

Chapter 1

Introduction

1.1 Motivation

Creating a personal virtual face on computers is still inconvenient in today's world. We are more frequently using personal computers to interact with our friends or give commands to computers. Being able to create virtual faces on computers and use them in daily communication will necessarily make people feel more comfortable and friendly to one another.

There are many types of virtual faces: 2D or 3D, photorealistic or non-photorealistic, etc. To produce a realistic animated 3D virtual face is still expensive today. For some applications, realism is not the only major importance. A well-expressed facial expression makes the conversation livelier and lets people know natural and extended meanings.

A cartoon-like virtual face can be used to reflect facial expressions including emotions and lip movements. In the past studies, cartoon faces were made by computer graphics techniques to show one's facial expressions.

There are more advantages in using non-photorealistic cartoon faces than using photorealistic ones, as described in the following.

- (1) There is more freedom in designing stylish cartoon faces. We can design the shapes and positions of each component, such as paths, regions, etc. We can even choose at will the stroke types or colors of each component.
- (2) Expressions can be modified and exaggerated by relocating predefined feature

points. That is, we can easily change expressions by adjusting the input parameters of the feature point positions.

- (3) Last but not least, cartoon faces are more interesting but may be created with less data. Personalized cartoon faces make themselves more meaningful for representing people. We can easily preserve them ourselves or share them with others.

However, a ten-minute animated cartoon needs at least 14400 frames. If we deal with them frame by frame, it will waste lots of time. We want to “control” our cartoon faces as easily as possible.

Computer facial animation has been developed for more than 30 years. It aims at modeling a virtual face to freely show facial expressions. Real-time motion capture techniques for this purpose have been studied for several years. Generally, by such techniques if one wants to make a penetrating real-time observation of his/her facial expressions to control a virtual face, he/she must put some markers on his/her face and then dynamically track them by sensors. After mapping the tracked markers to the virtual face, corresponding facial expressions can be created. However, this approach is usually complicated and expensive, and is inappropriate for general uses.

To demonstrate the feasibility of low-cost virtual face creation, we may use web-cameras to capture human faces. After detecting facial feature regions, facial feature points should be extracted and related parameters computed. Then cartoon faces can be created automatically by modifying the parameters. To animate personal cartoon faces more realistically, some basic emotions also need be specified. Moreover, lip movements may be synthesized from speech, and expressions may be synthesized from a video sequence of the user’s face without adopting the above-mentioned technique of attaching markers on faces.

In the goal of this study is to design an automatic system for an animated cartoon

face with moving lips uttering a synchronized Mandarin speech. It is hoped that by the use of the system, virtual teachers, virtual service agents, and so on, may be designed conveniently and become popular and useful in our life someday.

1.2 Survey of Related Studies

Generally speaking, the work of automatic generation of talking cartoon faces includes two main phases: automatic personal face generation and computer animation with speech synchronization.

Ruttkay *et al.* [4] defined a set of facial animation components to build a non-photorealistic cartoon face. Each component has its own potential dynamical behavior to be used in the animation. The work focused on how to animate a cartoon face without creating a personal cartoon face. Furthermore, some methods were proposed to create a skeleton-based cartoon faces [5, 6]. And in Litwinowicz *et al.* [7], a way is provided to animate skeleton-based faces. PicToon, designed in [8], is a cartoon system which can be used to generate a personalized cartoon face from an input picture. An example-based method is taken to generate the sketch lines of an input picture. By stroke rendering, a stylish cartoon face is created. PicToon also provides an audio-visual mapping between a character's voice and lips.

In order to animate a cartoon face for speaking, Li *et al.* [9] proposed a method to animate cartoon faces not only for lip synthesis, but also derived from speech signals a series of cartoon templates based on emotions. For Mandarin speaking, Lin and Tsai [1] reduced 411 viseme parameters to 167 classes to animate a photorealistic virtual face. Visemes were clustered by articulation and mouth shapes. A time separation concept was used for mouth synthesis and audio synchronization.

Moreover, we want to “control” our cartoon face by input facial images in the sense that the created cartoon face is similar to an input face image. Some methods were proposed to animate a virtual face by attaching markers on faces [14, 15]. If one wants to track facial features from sequential images without any marker, he/she needs to use some image processing techniques [13]. Tsai [2] proposed a moment-preserving thresholding technique. An effective eye-pair detection method was proposed in [10]. And Grinias *et al.* [11] provided a method to detect faces.

1.3 Overview of Proposed Method

An overview of the proposed approach is described in this section. First, some definitions of terms used in this study are introduced in Section 1.3.1. And several assumptions made for this study are listed in Section 1.3.2. Finally a brief description of the proposed method is given in Section 1.3.3.

1.3.1 Definitions of Terms

The definitions of several terms used later in this thesis are listed as follows.

(1) **Neutral Face:** MPEG-4 specifies some conditions for a head in its *neutral state* [12] as follows:

- Gaze is in the direction of the Z axis.
- All face muscles are relaxed.
- Eyelids are tangent to the iris.
- The pupil is one third of the iris diameter.
- The lips are in contact.
- The line of the lips is horizontal and at the same height of lip corners.

- The mouth is closed and the upper teeth touch the lower ones.
- The tongue is flat and horizontal with the tip of the tongue touching the boundary between the upper and lower teeth.

In the proposed system, a face with a normal expression is called a neutral face.

- (2) **Neutral Facial Image:** A neutral facial image is an image with a large frontal neutral face in it.
- (3) **Sequential Facial Images:** Sequential facial images are video sequences of a user's speaking face. One of these facial images is assigned to be a neutral facial image.
- (4) **Facial Features:** In the proposed system, we are concerned about several facial features. They are hair, face, eyebrows, eyes, nose, mouth, and ears of each facial image.
- (5) **Facial Expression:** A facial expression is a facial aspect or vocal intonation indicative of feeling. Here, Facial expressions include emotions and lip movements.
- (6) **Phoneme:** A phoneme is a basic enunciation of a language. For example, ㄅ, ㄆ, ㄇ are phonemes in Mandarin.
- (7) **Syllable:** A syllable consists of phonemes. For example, ㄅㄆ, ㄆㄇ are syllables in Mandarin.
- (8) **Visemes:** A viseme is the visual counterpart of a syllable.
- (9) **Speech Analyzer:** A speech analyzer takes a speech file and a script file as input for speech recognition to get timing information of each syllable.
- (10) **Key Frame:** A key frame can be any frame with the timeline feature in the animation at which you can precisely control the appearance of a cartoon face.

1.3.2 Assumptions

In real cases, it is complicated to detect faces in a disordered environment. In order to reduce the complexity of our work, a few assumptions and restrictions are made in this study. They are described as follows.

- (1) Colors of skin, hair, and background are not mixed.
- (2) Frontal lights were used in the environment with no shadow or light shadow behind a head.
- (3) A large frontal face is located in the center of each captured image.
- (4) The skew angle of the face does not exceed $\pm 10^\circ$.
- (5) The face position and the skew angle of each sequential facial image are almost fixed.

1.3.3 Brief Descriptions of Proposed Method

An overall procedure of the proposed method is illustrated in Fig. 1.1.

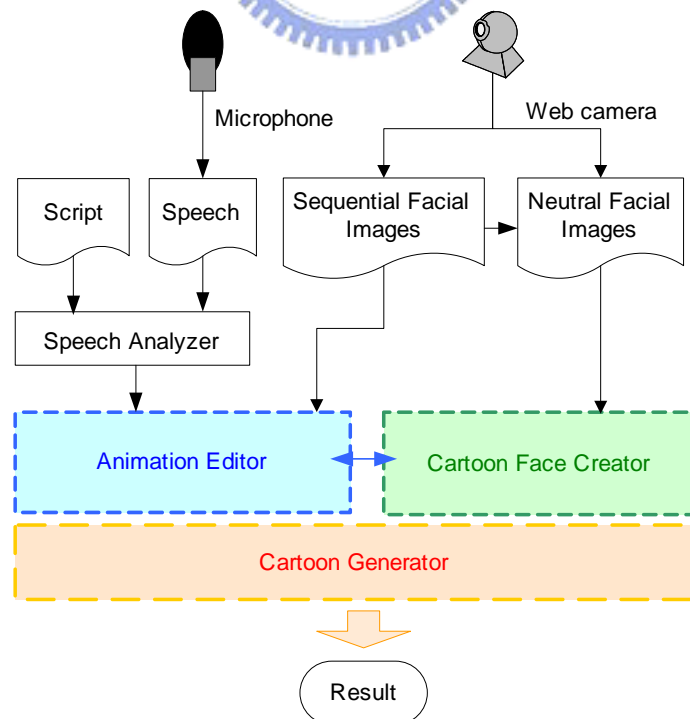


Fig. 1.1 A flowchart of proposed system.

The proposed system for talking cartoon face generation includes three major parts: a cartoon face creator, an animation editor, and a cartoon generator. The cartoon face creator creates neutral cartoon faces from neutral facial images. The animation editor manages animation processes for cartoon faces. Finally, the cartoon generator renders the style of cartoon faces.

We use a web camera and a microphone to get a single neutral facial image and a video clip of the user's speaking face. If one chooses to capture a single neutral facial image, a personal cartoon face will be created by the proposed cartoon face creator. If one wants to animate it, he/she may specify emotions or uses a speech file and a script file as input to synthesize lip movements by the animation editor. Finally, the cartoon generator will output a file of an animated talking cartoon face to the user.

Alternatively, one may use a video clip as input. After choosing one of the sequential facial images as a neutral facial image to create a neutral cartoon face, the animation editor will automatically track facial features of the remaining facial images to animate the cartoon face. Finally, the cartoon generator will output a file of an animated talking cartoon face which looks like the input video clip.

An illustration of the environment setup adopted in this study is shown in Fig. 1.2.

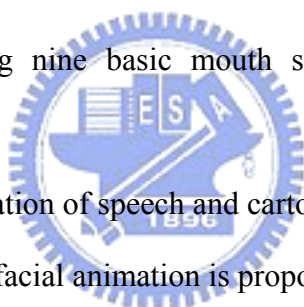


Fig. 1.2 Environment setup of this study.

1.4 Contributions

Some major contributions of this study are listed as follows.

- (1) A system for integration of creating and animating personal cartoon faces is designed.
- (2) An effective hierarchical bi-level thresholding method for extracting the face regions, hair regions, and background regions is proposed.
- (3) An effective knowledge-based method for detecting facial features is proposed.
- (4) An effective method for tracking facial features is proposed.
- (5) An adaptive method for simulating graphic-based personal cartoon faces is proposed.
- (6) A method of composing nine basic mouth shapes to synthesize Mandarin visemes is included.
- (7) A method for synchronization of speech and cartoon faces is proposed.
- (8) A method for smoothing facial animation is proposed.
- (9) An editable and opened vector-based XML language of W3C (World Wide Web Consortium) standard-SVG (Scalable Vector Graphics) is effectively used in the proposed system.



1.5 Thesis Organization

The remainder of this thesis is organized as follows. In Chapter 2, a more detailed overview of the proposed method is described to introduce the following chapters. In Chapter 3, a method of extracting facial feature regions is described. And then, a method of extracting facial feature points is described in Chapter 4. Up to Chapter 4, personal cartoon faces were created.

The methods we adopt for rendering a cartoon face from single facial image and from sequential facial images for synchronizing it with a speech file are described in Chapter 5 and Chapter 6, respectively. A final integration process using an application program SVG (Scalable Vector Graphics) and some experimental results are given in Chapter 7. Finally, conclusions and some suggestions for future works are included in Chapter 8.



Chapter 2

Overview of Proposed Method for Talking Cartoon Face Generation

2.1 System Organization

In this chapter, a more detailed overview of proposed system is described. The three major parts of the system, a cartoon face creator, an animation editor, and a cartoon generator, will be introduced further.

The cartoon face creator is used to create a personal cartoon face, with which we can detect the colors, geometries, and shapes of an input neutral facial image for cartoon face creation. If the user chooses a video clip as input, the cartoon face creator gets the first frame from the video sequence as a neutral facial image. In order to create a personal cartoon face, three main steps are included. The first is extraction of facial feature regions, which is applied to achieve a coarse detection of facial feature regions. The second step is extraction of facial feature points, which is expected to locate feature points for cartoon face drawing. The last step is definition of basic facial expression parameters for use in face animation.

The animation editor is used to deal with the animation of neutral cartoon faces. When a user chooses a single neutral facial image as input, the animation editor supports two kinds of processes: (1) synthesis of lip movements with synchronization with a speech file in accordance with a script file, and (2) assignment of emotions. Alternatively, if a user chooses a video clip as input, the animation editor tracks facial

features of the remaining frames to deal with the animation of the neutral cartoon face.

The cartoon generator is used to handle the final integration process from the views of both the temporal domain and the spatial domain using an application program SVG (Scalable Vector Graphics). From the temporal domain, the cartoon generator deals with animation and combines the speech file to make it synchronize with animation; and from the spatial domain, it controls the looks of a cartoon face. A flowchart of the organization and the animation generation stages of the proposed system are illustrated in Fig. 2.1.

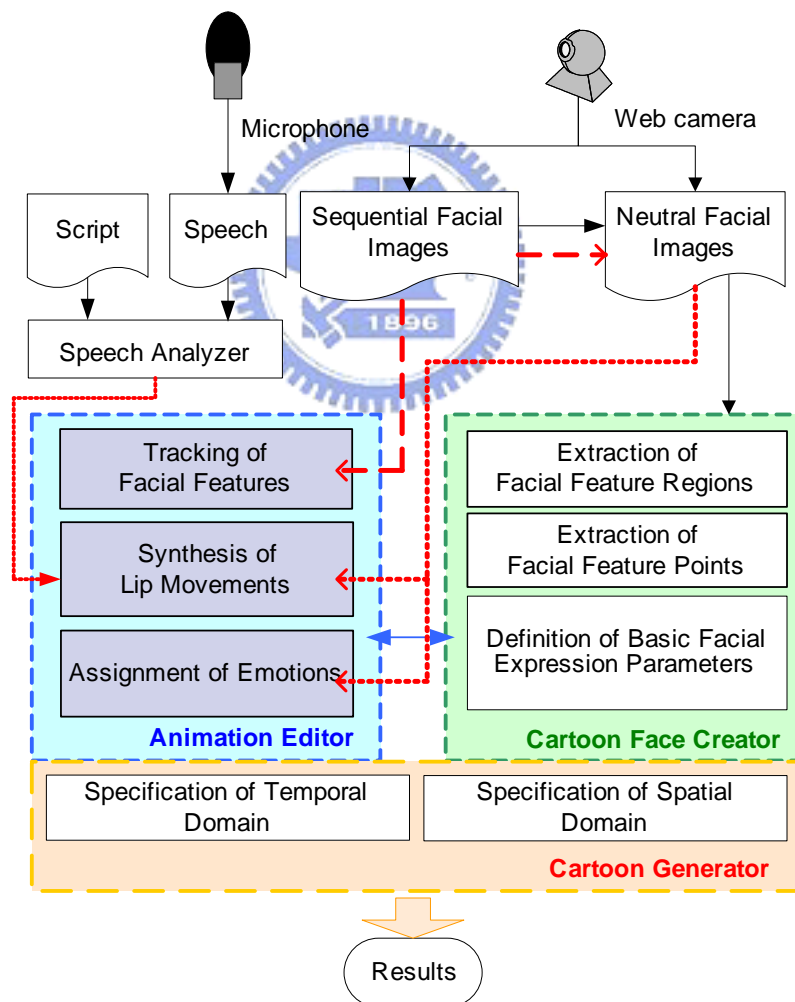


Fig. 2.1 System organization and generation stages.

2.2 Uses of Facial Features, Control Points, and Key Frames

Terms used in the cartoon face creator includes facial features (including facial feature regions and facial feature points) and control points. In order to animate cartoon faces, the use of key frames helps the animation editor to handle the animation.

Uses of facial feature regions and facial feature points are illustrated in Section 2.2.1 and 2.2.2, respectively. In Section 2.2.3, the use of control points is described. The concept of key frame use is illustrated in Section 2.2.4.

2.2.1 Facial Feature Regions

The use of facial feature regions gives us the information about the positions and ranges of facial features. Because each facial feature has its own characteristics, we have to extract its regions out for individual detailed detection. As shown in Fig. 2.2(a), the background regions, hair regions, and face regions are first separated. After knowing where the face regions are, other facial feature regions can be extracted using a knowledge-based technique, as shown in Fig. 2.2(b). A more detailed method to extract facial feature regions will be described in Chapter 3.

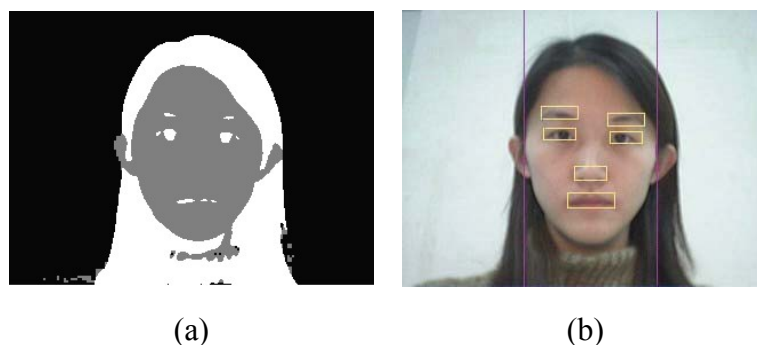


Fig. 2.2 Facial feature regions used in the proposed system. (a) Background, hair, and face regions. (b) Eyebrows, eyes, nose, and mouth regions.

2.2.2 Facial Feature Points

The use of facial feature points helps us to draw a neutral cartoon face. The use of facial feature points also has the capability to condense a large amount of raw data into a few meaningful points. By reassigning locations to or dragging these feature points, we can deform cartoon faces directly. In the proposed system, 72 facial feature points are used to draw a cartoon face. A result of extraction of facial feature points is shown in Fig. 2.3. In Chapter4, a more detail description of the proposed extraction method will be described.

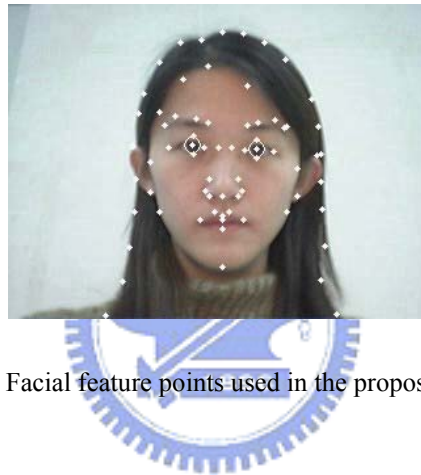


Fig. 2.3 Facial feature points used in the proposed system.

2.2.3 Control Points

Some facial feature points are assigned to be control points for animating particular facial features. For example, if we want to deform mouth shapes, we only have to change points 8.9, 8.4, 8.2, and 8.3, as shown in Fig. 2.4. The other feature points of the mouth, like 8.1, 8.10, 2.2, 2.3, will move automatically by a pre-defined rule.

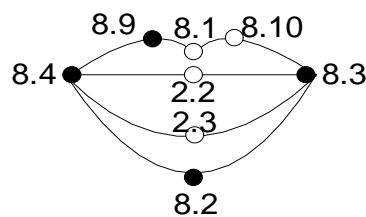


Fig. 2.4 Feature points of mouth.

2.2.4 Key Frames

A timeline was used as a basic structure for animation. If we want to have a 1-minute cartoon with 25 frames per second, we must preserve at least 1500 frames.

The most important part of animation is dealing with key frames. By knowing when and how a facial expression shows, the positions of the control points are defined in corresponding key frames. And then, after applying an interpolation technique to compute the remaining frames between key frames, cartoon faces can be animated, as shown in Fig. 2.5.

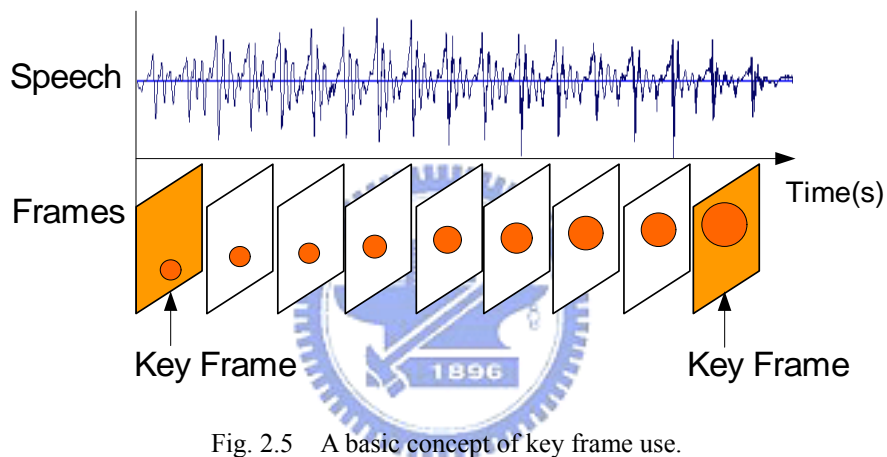


Fig. 2.5 A basic concept of key frame use.

2.3 Talking Cartoon Face Generation from Single Images

In this section, the basic idea of talking cartoon face generation from single images is illustrated in Section 2.3.1. And an overall generation process is described in Section 2.3.2.

2.3.1 Basic Idea

If a user wants to generate a talking cartoon face from single images, he/she must take a neutral facial image as input data first. The cartoon face generator will then process the image to create a neutral cartoon face. In the meanwhile, basic facial expression parameters must also be defined for the animation editor to use.

Because there is only one input image, we can just animate the cartoon face by synthesizing moving lips and emotions. To synthesize moving lips, an additional speech analyzer is used to extract timing information of syllables from a speech file and a script file. A syllabus may be composed of several basic mouth shapes from adjusting the basic facial feature parameters. By assigning key frames with corresponding mouth shapes, cartoon faces can be animated.

To synthesize emotions, the user can assign the talking cartoon face a freely specified emotion. By reassigning locations to the feature points of eyebrows and eyes, the cartoon faces may appear to be more different.

Finally, the cartoon generator will render the style of cartoon faces and synchronize the speech file with the animation. A file of an animated talking cartoon face with moving lips uttering a synchronized Mandarin speech can be generated as the final output.

More detailed descriptions of the involved steps of the method will be described in Chapter 5.

2.3.2 Generation Process

Fig. 2.6 illustrates a flowchart of the stages of talking cartoon face generation from single images. First, a neutral facial image is used as input to the cartoon face creator. After extracting facial feature regions and facial feature points, control points

then are assigned to define basic facial expression parameters. After that, a neutral cartoon face is created.

Second, the animation editor is used to deal with the animation of the neutral cartoon face. A speech file and a script file are used to synthesize moving lips. Also, a user can freely specify emotions. Then, the animation editor will combine these two processes.

Finally, the cartoon generator outputs a file of an animated talking cartoon face.

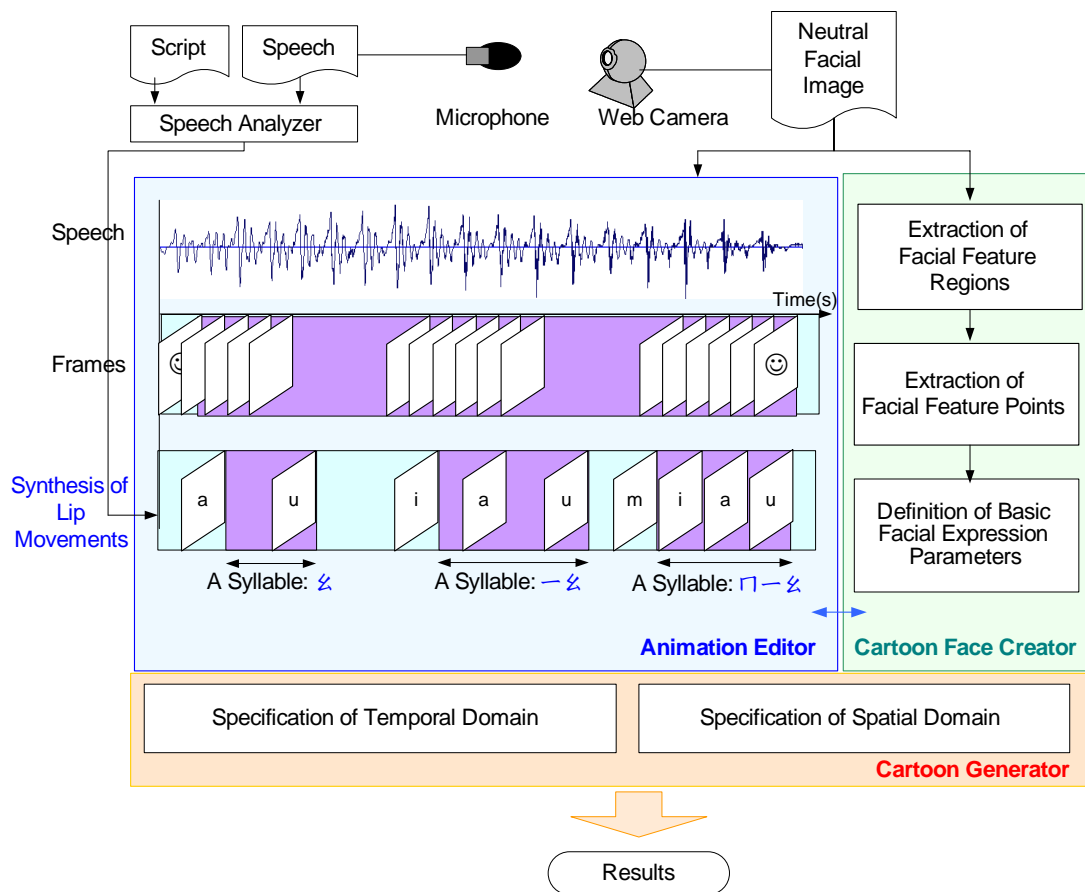


Fig. 2.6 Stages of talking cartoon face generation from single images.

2.4 Talking Cartoon Face Generation from Image Sequences

In this section, the basic idea of talking cartoon face generation from image sequences is illustrated in Section 2.4.1. And an overall generation process is described in Section 2.4.2.

2.4.1 Basic Idea

Alternatively, if a user wants to generate a talking cartoon face from a given image sequence, he/she must take a video clip of the user's talking face as input. The video clip will be separated into a sequence of facial images and a speech file. The cartoon face creator will then select the first frame as a neutral facial image to create a neutral cartoon face. Basic facial expression parameters will be defined for the use of adjusting the neutral facial image to get face tracking results.

The animation editor will then process the remaining frames to make the neutral cartoon face act like the face in the video. Every frame of the video sequence will be regarded as a key frame. In the proposed system, an effective tracking method is proposed to track the eyes, and mouth. By assigning the tracking information as key frames in a proper sequence, cartoon faces will be animated.

Finally, the cartoon generator will recombine the speech file with the animated cartoon face, and then output a file of an animated talking cartoon face in the sense that the created cartoon face is similar to the user's face.

More detailed descriptions of the involved steps of the method will be described in Chapter 6.

2.4.2 Generation Process

Fig. 2.7 illustrates a flowchart of the stages of talking cartoon face generation from image sequences. First, the first frame of a video sequence selected to be a neutral facial image as input to the cartoon face creator. After extracting the facial feature regions and facial feature points, control points will then be assigned to define basic facial expression parameters. After that, a neutral cartoon face is created.

Second, the animation editor will then extract facial features from the remaining frames to animate the cartoon face. Finally, the cartoon generator recombines the speech file and the animated cartoon face to yield as output a file of an animated talking cartoon face.

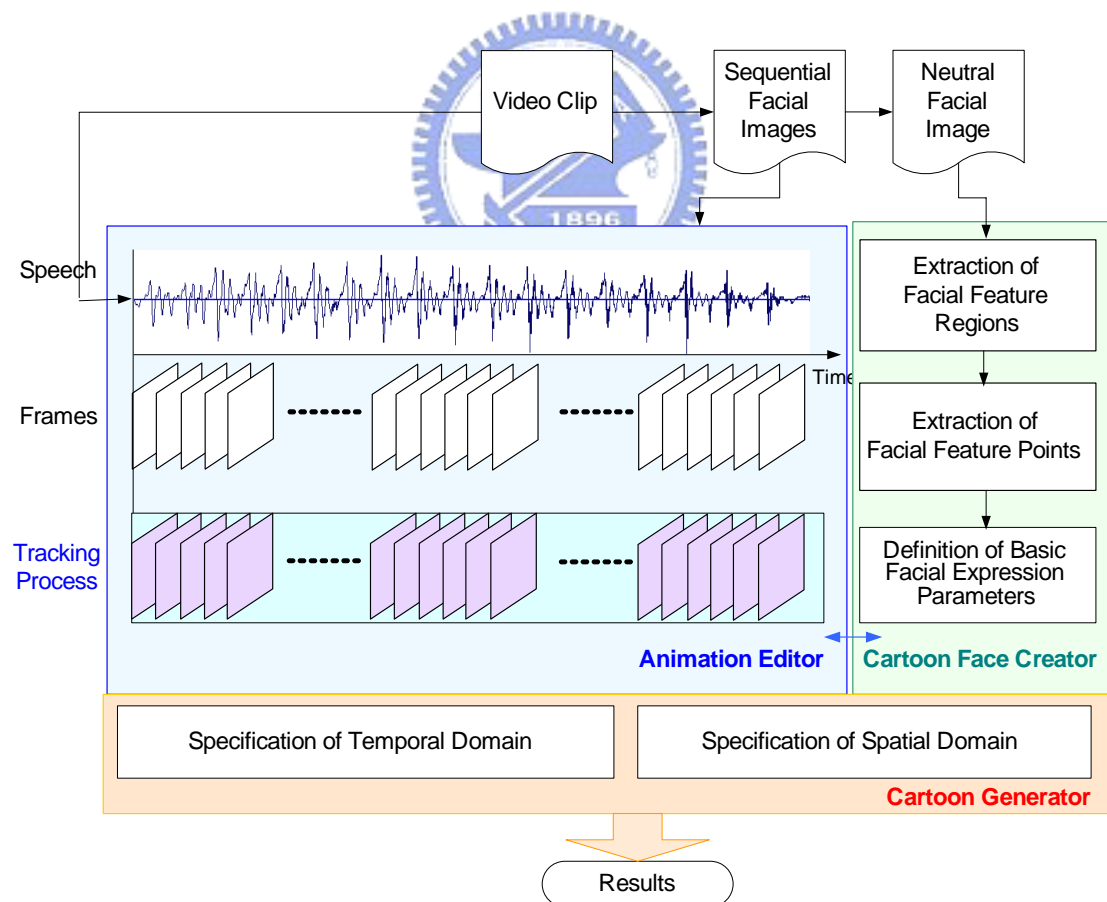


Fig. 2.7 Stages of talking cartoon face generation from sequential images.

Chapter 3

Extraction of Facial Feature Regions

3.1 Introduction

In this study, a method is proposed to extract facial feature regions. First, a hierarchical bi-level thresholding method is proposed to extract the background regions, hair regions, and face regions in a given face image. By applying a region refinement method, a final background region, a hair region, and a face region are obtained. In the meanwhile, based on Chan *et al.* [10], a pair of eye regions is detected for subsequent use. A review of Chan's method is described in Section 3.2. A more detailed description of the proposed method is given in Section 3.3.

Furthermore, a knowledge-based technique was applied to extract other facial feature regions, including eyebrow regions, eye regions, a nose region, and a mouth region. A more detailed description of the method is given in Section 3.4.

Finally, a method for detection of cheek boundaries and ear regions is proposed in Section 3.5. An overall illustration of the proposed methods with experimental results is shown in Fig. 3.1.

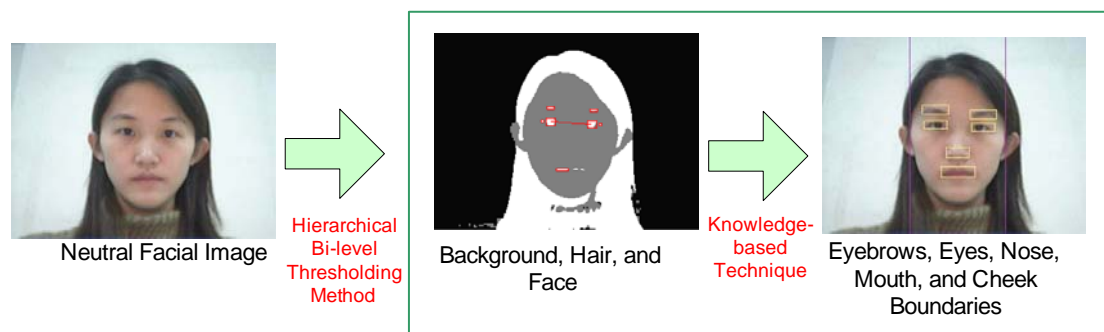


Fig. 3.1 An overall illustration result of extraction of facial feature regions.

3.2 Review of Eye-pair Detection Method

Chan *et al.* [10] proposed a fast frontal face detection method that pairs probable eye regions together to extract face areas. Fig. 3.2 illustrates the individual stages of the process.

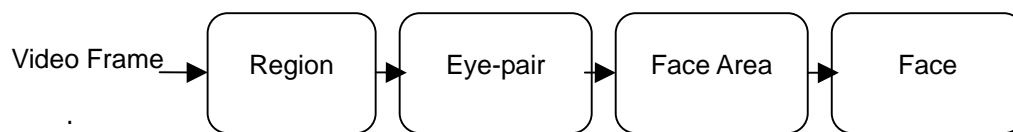


Fig. 3.2 Stages of the fast frontal face detection method by Chan *et al.* [10].

In Chan's method, the initial stage is to acquire an image and segment it into regions. Only the first significant bits in the red, green, and blue channels are used to reduce color data. Regions are then extracted by visiting every pixel as a seed pixel and flood filling surrounding areas that have the same values as that of the current seed pixel. In order to reduce the computational complexity, regions are filtered according to a set of rules related to the regions' heights, widths, etc. Next, the eye-pair generation process attempts to pair the extracted regions together to check if they have the potential to be an eye-pair. Finally, in the face area extraction stage, a square region is extracted for final face verification which is based on a neural network technique. An illustration of the square region is shown in Fig. 3.3.

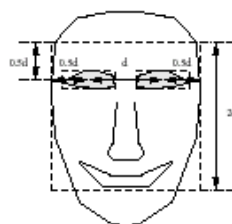


Fig. 3.3 The square region proposed in Chan *et al.* [10].

3.3 Extraction of Background, Hair, and Face Regions

The HSI (Hue, Saturation, and Intensity) color model is used in this study to handle images because it is closely related to the way in which humans perceive colors. The hue component describes the color itself in the form of an angle between 0 to 360 degrees, as shown in Fig. 3.4. The range of the S component is [0, 1]. And the intensity range is between [0, 1], where 0 means black, and 1 means white.

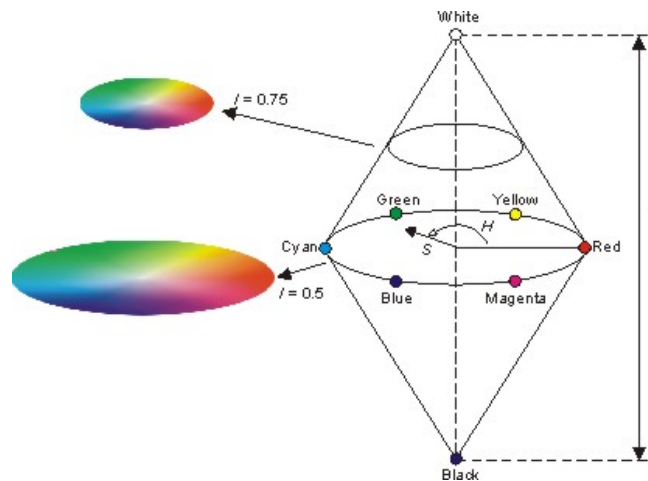


Fig. 3.4 HSI color space.

The proposed hierarchical bi-level thresholding method mentioned previously is described in Section 3.3.1. The 1st level of this hierarchical bi-level thresholding method is applied in the intensity channel to extract dark regions and light regions. The detail is described in Section 3.3.2. An optimal eye-pair generation method based on dark regions is then proposed and described in Section 3.3.3. Based on light regions, the 2nd level of the above-mentioned hierarchical bi-level thresholding method is applied in the hue channel to extract face regions and background regions, and the detail is described in Section 3.3.4. Finally, Section 3.3.5 describes the

previously-mentioned final region refinement method to get a final face region and a hair region.

3.3.1 Proposed Hierarchical Bi-level Thresholding

The 1st level bi-level thresholding of the proposed hierarchical bi-level thresholding method is used to separate dark regions (including eye regions and hair regions) and light regions (including background regions and face regions) in the intensity channel. Two of the dark regions are detected to be an optimal eye-pair. Then the remaining regions are regarded as hair regions.

Since the light regions consist of face regions and background regions, a 2nd level bi-level thresholding is applied in the hue channel to separate them apart. After applying the region refinement method, a final face region and a hair region are obtained. On the contrary, the background region is eliminated. A flowchart of using the proposed hierarchical bi-level thresholding method to extract the above mentioned three kinds of facial feature regions is shown in Fig. 3.5.

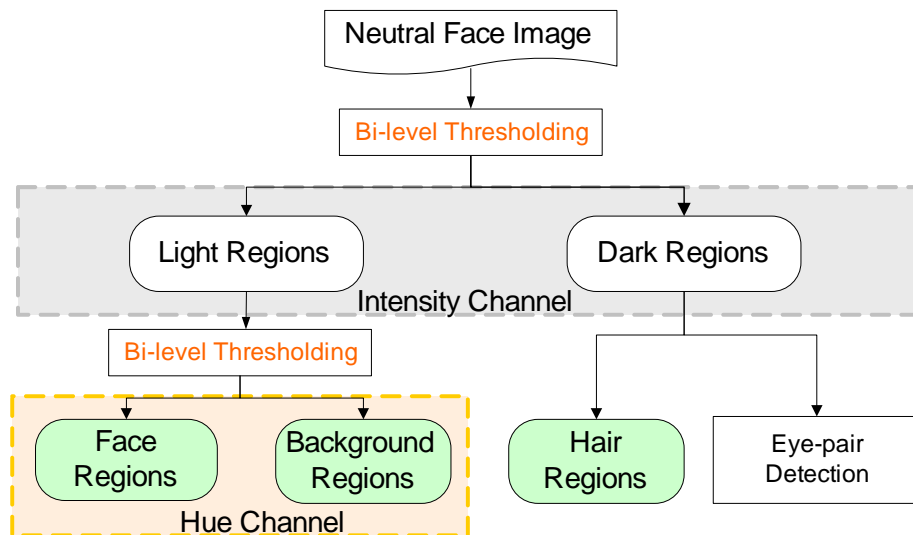


Fig. 3.5 Flowchart of hierarchical bi-level thresholding method.

3.3.2 1st Level Bi-level Thresholding in Intensity Channel

As defined in Chapter1, a neutral facial image is an image with a large frontal neutral face in it. Therefore, we assume that a central rectangle can be used to collect the color information of a user's face. The following algorithm implements the previously-mentioned 1st level thresholding to get a binary image and two sets of different kind of regions.

Algorithm 4.1. *Bi-level thresholding in intensity channel.*

Input: A neutral facial image N and a central rectangle $Rect$ with size $m \times n$.

Output: A binary image B_1 , a set of dark regions D , and a set of light regions L .

Steps:

1. Transform N into the intensity channel to get $I(x, y)$, where $I(x, y)$ denotes the intensity value at pixel (x, y) .
2. Compute a mean value m_1 and a standard deviation value s_1 in $Rect$ by the following way:

$$m_1 = \frac{1}{m \times n} \sum_{x, y \in Rect} I(x, y); \quad (3.1)$$

$$s_1 = \left(\frac{1}{m \times n - 1} \sum_{x, y \in Rect} (I(x, y) - m_1)^2 \right)^{1/2}. \quad (3.2)$$

3. Compute a threshold value t_1 by:

$$t_1 = m_1 - s_1. \quad (3.3)$$

4. Threshold I with t_1 to get a binary image $B_1(x, y)$ by the following equation:

$$B_1(x, y) = \begin{cases} 1, & \text{if } I(x, y) > t_1 \\ 0, & \text{if } I(x, y) \leq t_1 \end{cases}. \quad (3.4)$$

Thus, pixels of $B_1(x, y)$ labeled 1 correspond to light pixels, whereas pixels labeled 0 correspond to dark pixels.

5. Find pixels of $B_1(x, y)$ with values of 0, use them as seeds, and apply region growing to extract a set of dark regions D .
6. Find pixels of $B_1(x, y)$ with values of 1, use them as seeds, and apply region growing to extract a set of light regions L .

An experimental result of the above algorithm is shown in Fig. 3.6.

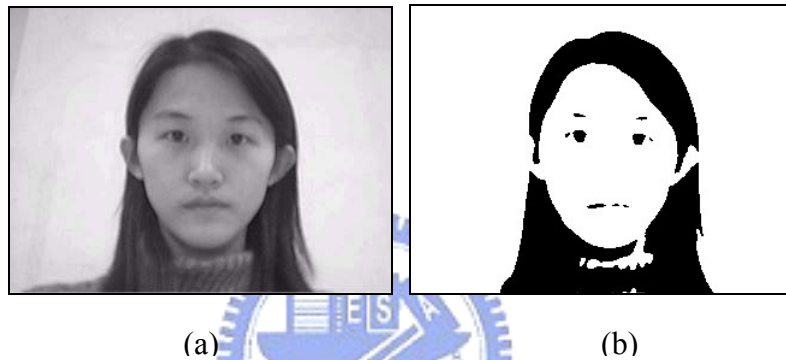


Fig. 3.6 An experimental result of bi-level thresholding in intensity channel.

(a) Neutral facial image in intensity channel. (b) Binary image B_1 .

3.3.3 Optimal Eye-Pair Detection

Since the set of dark regions D generated by Algorithm 4.1 consists of two kinds of facial feature regions (including eye regions and hair regions), we need a method to extract two eye regions first and regard the remaining regions as hair regions. Since Chan's method [10] for eye-pair generation will output one or more probable eye-pairs, a method is proposed to select an optimal eye-pair, as described in the following algorithm.

Algorithm 4.2. *Optimal eye-pair detection.*

Input: A set of dark regions D generated by Algorithm 4.1

Output: Eye-pair regions E_j and E_k , where j, k are the region indices of an eye-pair.

Steps:

1. Apply Chan's method [10] to D to conduct eye-pair generation to get a probable set of eye-pair regions E . Let E_i denote the i^{th} eye-pair in E .
2. Compute the areas of the two regions, denoted by $E_{i,1}$ and $E_{i,2}$, of each eye-pair E_i in the following way, where $Area_{i,1} \geq Area_{i,2}$, $Area_{i,1}$ and $Area_{i,2}$ are normalized:
 - $Area_{i,1} = E_{i,1}.width \times E_{i,1}.height$;
 - $Area_{i,2} = E_{i,2}.width \times E_{i,2}.height$.

3. Compute a score of each eye-pair E_i by:

$$Score_i = \frac{Area_{i,1} \times Area_{i,2} - S_i^2}{2}, \text{ where } S_i = \frac{Area_{i,1}}{Area_{i,2}} \quad (3.5)$$

For the largest $Score_i$, set $E_j = E_{i,1}$ and $E_k = E_{i,2}$.

An experimental result of the above algorithm is shown in Fig. 3.7.

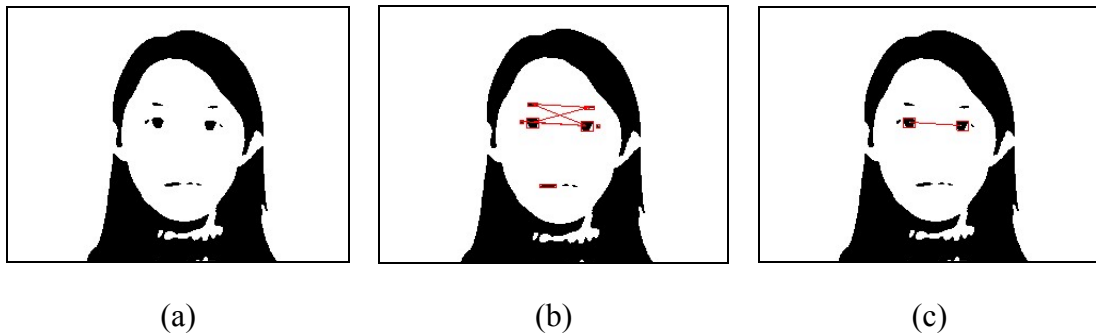


Fig. 3.7 An experimental result of optimal eye-pair detection. (a) Binary image B_1 . (b) Probable eye-pairs generated by Chan's method. (c) An optimal eye-pair.

3.3.4 2nd Level Bi-level Thresholding in Hue Channel

Since the set of light regions L generated by Algorithm 4.1 consist of two kinds of facial feature regions (including face regions and background regions), in order to separate them apart, the 2nd level of the proposed hierarchical bi-level thresholding method is applied in the hue channel to achieve the goal.

Algorithm 4.3. *Bi-level thresholding in hue channel.*

Input: A neutral facial image N , a binary image B_1 created by Algorithm 4.1, and a central rectangle $Rect$ with size $m \times n$.

Output: A binary image B_2 , a set of face regions F , and a set of background regions Bg .

Steps:

1. Transform N into the hue channel and normalize it into the range of $[-127, 128]$ to get H . Let $H(x, y)$ denote the hue value of pixel (x, y) .
2. Compute a mean value m_2 and a standard deviation value s_2 in $Rect$ by the following way:

$$m_2 = \frac{1}{m \times n} \sum_{x, y \in Rect} H(x, y); \quad (3.6)$$

$$s_2 = \left(\frac{1}{m \times n - 1} \sum_{x, y \in Rect} (H(x, y) - m_2)^2 \right)^{1/2}. \quad (3.7)$$

3. Threshold H by applying the Euclidean distance by the following equations:

$$d_H(x, y) = |H(x, y) - m_2|; \quad (3.8)$$

$$B_2(x, y) = \begin{cases} 1, & \text{if } d_H(x, y) > s_2 \\ 0, & \text{if } d_H(x, y) \leq s_2, B_1(x, y) = 1 \end{cases}. \quad (3.9)$$

Thus, the pixels of $B_2(x, y)$ labeled 1 correspond to background pixels,

whereas pixels labeled 0 correspond to face pixels.

4. Find pixels of $B_2(x,y)$ with values of 0, use them as seeds, and apply region growing to extract a set of face regions F .
5. Find pixels of $B_2(x,y)$ with values of 1, use them as seeds, and apply region growing to extract a set of background regions Bg .

An experimental result of the above algorithm is shown in Fig. 3.8.

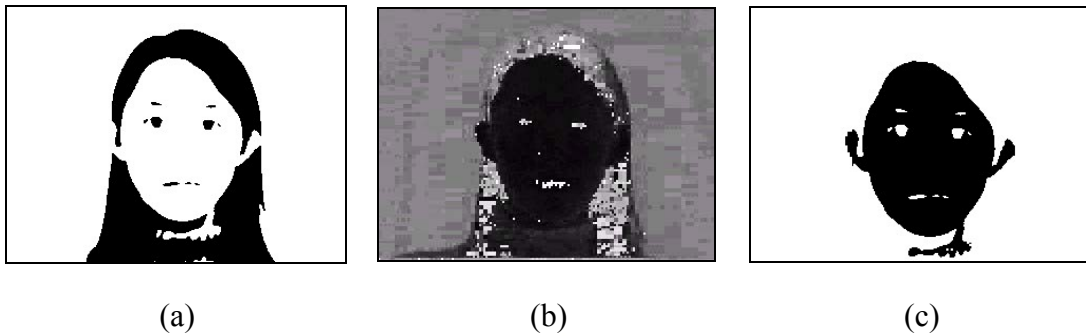


Fig. 3.8 An experimental result of bi-level thresholding in hue channel. (a) Binary image B_1 .

(b) Neutral facial image in hue channel. (c) Binary image B_2 .

3.3.5 Region Refinements

After applying the proposed hierarchical bi-level thresholding method, four sets of regions including dark regions D , light regions L , face regions F , and background regions Bg are extracted. Besides, an optimal eye-pair detection method is used to generate two eye regions E_j and E_k . Fig. 3.9 shows the above mentioned using four sets of regions and two eye regions.

In order to obtain final face regions and hair regions, a region refinement process is proposed to achieve the goal.

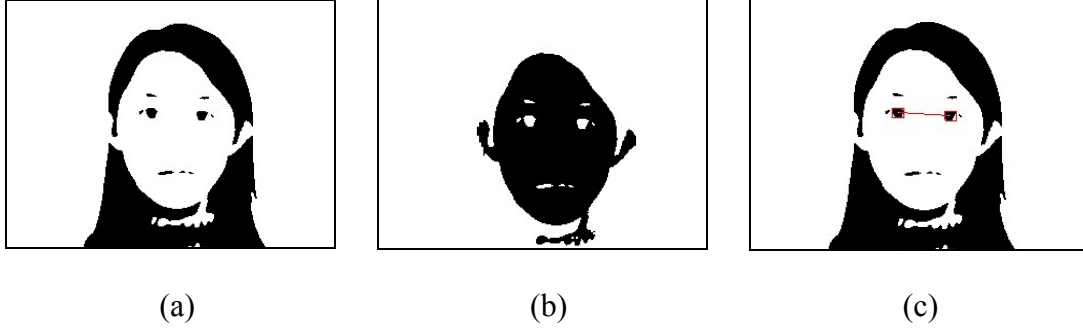


Fig. 3.9 Four sets of regions and two eye regions. (a) Dark regions D (black) and light regions L (white) of B_1 . (b) Face regions F (black) and background regions (white) of B_2 . (c) Eye regions (red rectangles).

Algorithm 4.4. *Region refinement for face regions and hair regions.*

Input: A binary image B_1 , a binary image B_2 , a set of face regions F , and a set of background regions Bg .

Output: A face regions F_{final} , and a hair region H_{final} .

Steps:

1. Choose pixels at the boundary of B_2 having values of 1 to get a set of regions Bg' , with $Bg' \in Bg$.
2. Obtain a face region by the following equation:

$$F_{final} = F \cup (Bg - Bg') \quad (3.10)$$

3. Compute a new binary image B_3 by the following equation:

$$B_3(x, y) = B_1(x, y) \text{ AND } B_2(x, y), \quad (3.11)$$

where *AND* is a logic operator.

4. Find pixels at the boundary of B_3 with values of 1, use them as seeds, and apply region growing to get a set of regions H' .
5. Obtain a hair region by the following equation:

$$H_{final} = R - H', \text{ where } R \text{ represents the entire image region} \quad (3.12)$$

An experimental result of the above algorithm is shown in Fig. 3.10.

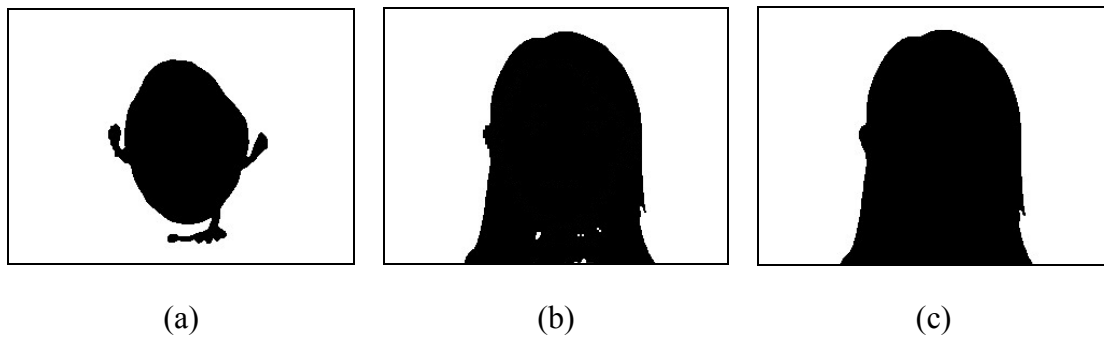


Fig. 3.10 An experimental result of region refinements for face regions and hair regions. (a) Face region (black). (b) Bi-level image B_3 . (c) Hair region (black).

3.4 Extraction of Facial Feature Regions



The proposed knowledge-based technique mentioned previously is based on a concept to estimate the positions and ranges of facial feature regions by the information about the detected eye-pair. The proposed method includes three basic processes. First, a more precise detection of a pair of eyeballs for the estimation of the positions and ranges of other facial features is described in Section 3.4.1. The second process is a series of preprocessing steps applied to the neutral facial image to get a binary edge image, which is described in Section 3.4.2. Finally, a knowledge-based estimation is used to detect the positions and ranges of eyebrow regions, eye regions, nose regions, and mouth regions, as described in Section 3.4.3.

3.4.1 Eyeball Detection for Position Estimation

In Chan *et al.* [10], a square region $2d \times 2d$ is proposed to cover the main facial features, where d is the Euclidean distance between the centers of the pair of eye regions. In Section 3.3.3, we have described how to detect a pair of regions as an eye-pair. However, it is still not precise enough for the use of the knowledge-base technique. For each region of the eye-pair, a circle with the maximum number of points in it is detected to be an eyeball. Additionally, information about the position and diameter of each eyeball is obtained. Some definitions are listed as follows:

- $Eyeball_{Left/Right}.x$ is the x position of the center of the left/right eyeball circle;
- $Eyeball_{Left/Right}.y$ is the y position of the center of the left/right eyeball circle;
- $Eyeball_{Left/Right}.radius$ is the radius of the left/right eyeball circle;
- $Eyeball_{Left/Right}.height$ is the height of the left/right eyeball circle;
- $EyeMid$ is the position (x, y) of the center between $Eyeball_{Left}$ and $Eyeball_{Right}$.

After computing a Euclidean distance d between two eyeballs (as shown in Fig. 3.11(a)), a $2d \times 2d$ square region is detected (as shown in Fig. 3.11(b)).

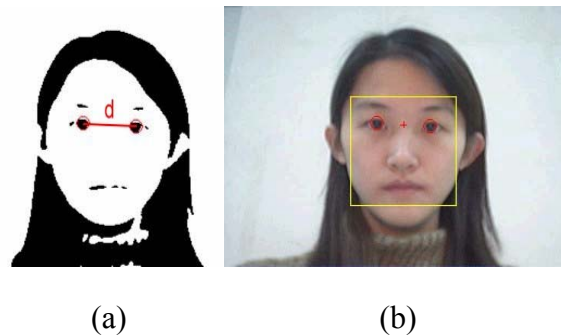


Fig. 3.11 A result of eyeball detection. (a) Eyeballs with a distance d between them. (b) A $2d \times 2d$ square region.

3.4.2 Knowledge-based Edge Detection by Local Thresholding

In the proposed system, the information of edge points is used to detect facial feature regions, including eyebrow regions, eye regions, nose regions, and mouth regions. A method is needed to generate a useable binary edge image, as described in the following algorithm.

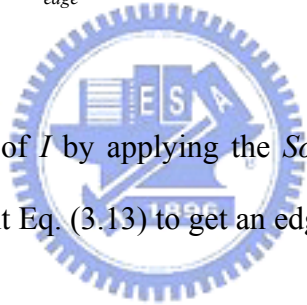
Algorithm 4.5. *Edge detection by local thresholding.*

Input: A grey-level image I generated by Step1 of Algorithm 4.1, and a value d defined in Section 3.4.1.

Output: A binary edge image B_{edge} .

Steps:

1. Detect the edges of I by applying the *Sobel operator*, as shown in Fig. 3.12, to implement Eq. (3.13) to get an edge image S .



| | | |
|----|----|----|
| -1 | -2 | -1 |
| 0 | 0 | 0 |
| 1 | 2 | 1 |

| | | |
|----|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

Fig. 3.12 Sobel operators.

$$S(x, y) = \left| (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \right| + \left| (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \right| \quad (3.13)$$

where z_5 denotes $I(x, y)$, z_1 denotes $I(x-1, y-1)$, and so on.

2. Mirror map the right half side of S in the $2d \times 2d$ rectangle range to another side according to the positions of two eyeballs.

3. Speculate three horizontal division lines, Div_{EE} (eyebrows to eyes), Div_{EN} (eyes to nose), and Div_{NM} (nose to mouth), by an experimental result. And then divide S into four parts.
 - $Div_{EE} = Eyeball_{Left}.y - Eyeball_{Left}.height / 2 - 1$
 - $Div_{EN} = EyeMid.y + d / 3$
 - $Div_{NM} = EyeMid.y + d$
4. Threshold each part of S by apply Tsai's moment-preserving thresholding method [2] to get the binary edge image B_{edge} .

An experimental result is shown in Fig. 3.13.

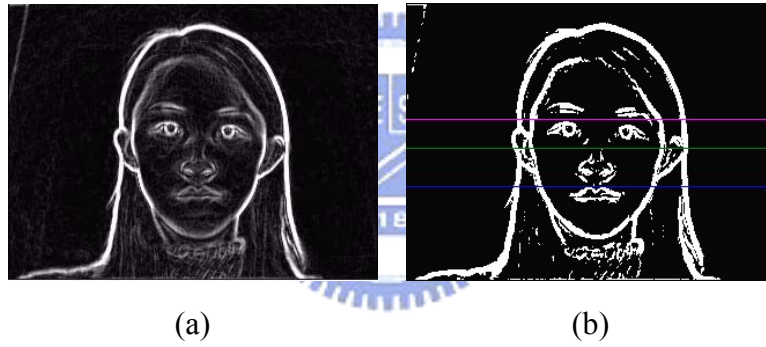


Fig. 3.13 A result of knowledge-based edge detection by local thresholding. (a) Sobel edge image S .
 (b) Binary edge image B_{edge} with three horizontal divisions shown on it.

3.4.3 Knowledge-based Facial Feature Extraction

The main idea of the proposed knowledge-based facial feature extraction method is to speculate an initial search region of each facial feature to extract a final region in B_{edge} by a region growing method. The extracted $2d \times 2d$ square range is used here to speculate the initial position and range of each facial feature region. The detailed rules for extracting facial feature regions in edge image B_{edge} are illustrated in the

following as an algorithm.

Algorithm 4.6. *Knowledge-based facial feature extraction.*

Input: Two eyeball regions (including $Eyeball_{Left}$ and $Eyeball_{Right}$ detected in Section 3.4.1), a position of $EyeMid$ defined in Section 3.4.1, a value d defined in Section 3.4.1, and three division lines (including Div_{EE} , Div_{EN} and Div_{NM} illustrated in Section 3.4.2).

Output: A final left eyebrow region $R_{Left_eyebrow}$, a final left eye region R_{Left_Eye} , a final nose region R_{Nose} , and a final mouth region R_{Mouth} .

Steps:

1. Speculate an initial search range of the left eye $R_{initial_{Left_Eye}}$ to extract the final left eye region R_{Left_Eye} in the following way:

$R_{initial_{Left_Eye}} = [P_1, P_2]$, where

$P_1 = (Eyeball_{Left}.x - Eyeball_{Left}.height, Eyeball_{Left}.y - 0.5 \times Eyeball_{Left}.height)$, and

$P_2 = (Eyeball_{Left}.x + Eyeball_{Left}.height, Eyeball_{Left}.y + 0.5 \times Eyeball_{Left}.height)$.

2. Speculate an initial search range of the left eyebrow $R_{initial_{Left_Eyebrow}}$ to extract the final left eyebrow region $R_{Left_Eyebrow}$ in the following way:

$R_{initial_{Left_Eyebrow}} = R_{Left_Eye}$ translated upper than the division line Div_{EE} .

3. Speculate an initial search range of the nose $R_{initial_{Nose}}$ to extract the final nose region R_{Nose} in the following way:

$R_{initial_{Nose}} = [P_3, P_4]$, where

$P_3 = (Eyeball_{Left}.x, Div_{EN})$, and

$P_4 = (Eyeball_{Right}.x, Div_{EN} + 0.7 \times (Div_{NM} - Div_{EN}))$.

4. Speculate an initial search range of the mouth $R_{initial_{Mouth}}$ to extract the final mouth region R_{Mouth} in the following way:

$R_{initial_{Mouth}} = [P_5 P_6]$, where

$P_5 = (Eyeball_{Left} .x, Div_{NL})$, and

$P_6 = (Eyeball_{Right} .x, Div_{NM} + d/3)$.

An illustration of the searching ranges represented as rectangles is shown in Fig.

3.14.

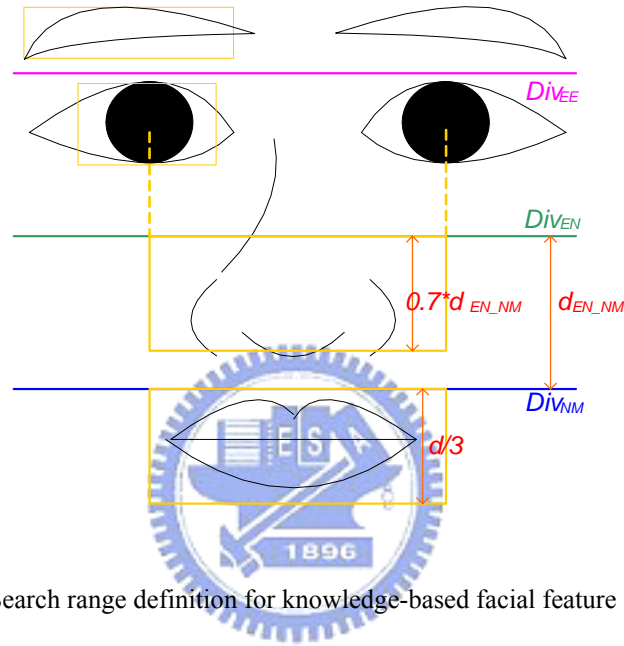


Fig. 3.14 Search range definition for knowledge-based facial feature extraction.

Some restrictions are also made in this study to avoid errors caused by noise, as described in the following:

- The boundaries of each final facial feature regions do not exceed the $2d \times 2d$ square range.
- The width boundaries of the final nose region and mouth region do not exceed the centers of eyeballs.
- A central vertical cut can be used for fine tuning to the final nose region and mouth region. A detailed method of finding a vertical cut of each region is described in the following algorithm.

Algorithm 3.7 *Central vertical cut detection of nose region and mouth region.*

Input: A grey-level image I generated by Step1 of Algorithm 4.1, a final nose region

R_{Nose} , and a final mouth region R_{Mouth} .

Output: A central vertical cut of R_{Nose} called L_{Nose} and a central vertical cut of R_{Mouth} called L_{Mouth} .

Steps:

1. Let the region R_{Nose} be denoted by $[(x_1, y_1), (x_2, y_2)]$.

2. Set $x = (x_1 + x_2)/2$.

3. Compute a width w with the following equation:

$$w = \frac{(x - x_1) + (x_2 - x)}{2}. \quad (3.14)$$

4. Set the left half region R_{Left} as $[(x-w, y_1), (x-1, y_2)]$, and the right half region R_{Right} as $[(x+1, y_1), (x+w, y_2)]$.

5. Compute a dissimilarity value D_x by the following equation:

$$D_x = \sum_{j=y_1}^{y_2} \sum_{i=1}^w |I(x-i, j) - I(x+i, j)|. \quad (3.15)$$

6. Shift x for the next detection in the following way.

6.1. Subtract 1 from x and go to Step 3.

6.2. Add 1 to x and go to Step 3.

7. Set the vertical line L_{Nose} equal to the x with a minimal D_x .

8. Repeat Steps 2 to 6 to get a vertical line L_{Mouth} .

An experimental result is shown in Fig. 3.15.

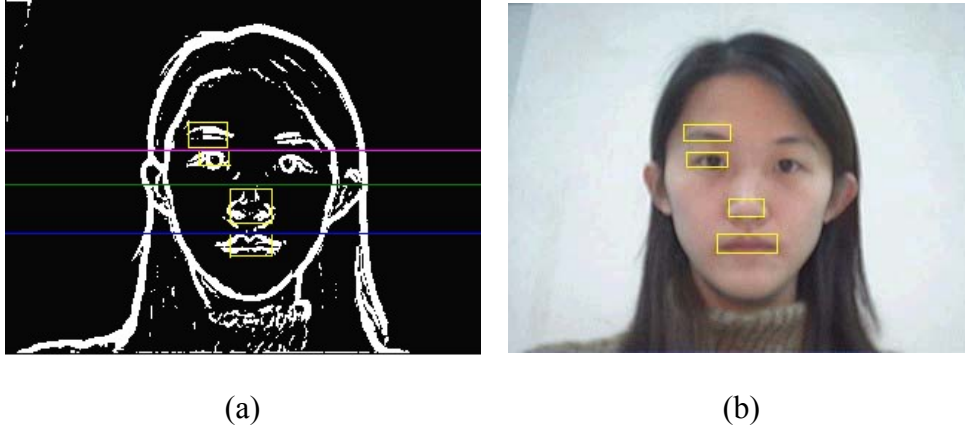


Fig. 3.15 A result of extraction of facial feature regions. (a) Speculated positions and ranges of facial features (yellow rectangle). (b) Final extraction results of facial feature regions.

3.5 Detection of Cheek Boundaries and Ear Regions

After extracting the above mentioned facial feature regions, there are still some features needed to be detected. In order to extract ear regions, cheek boundaries need to be detected first.

In order to detect cheek boundaries, a vertical projection method is applied to the set of face regions F_{final} . After mirror-mapping the right-side projection information to the left side, the proposed method also considers the symmetry characteristic of human faces. Finally, we find a position near the $2d \times 2d$ square range with a local minimum projection value as the desired cheek boundaries.

After detecting the position of cheek boundaries, the proposed method divides two parts of the set of face regions F_{final} as ear regions.

An experimental result is shown in Fig. 3.16.

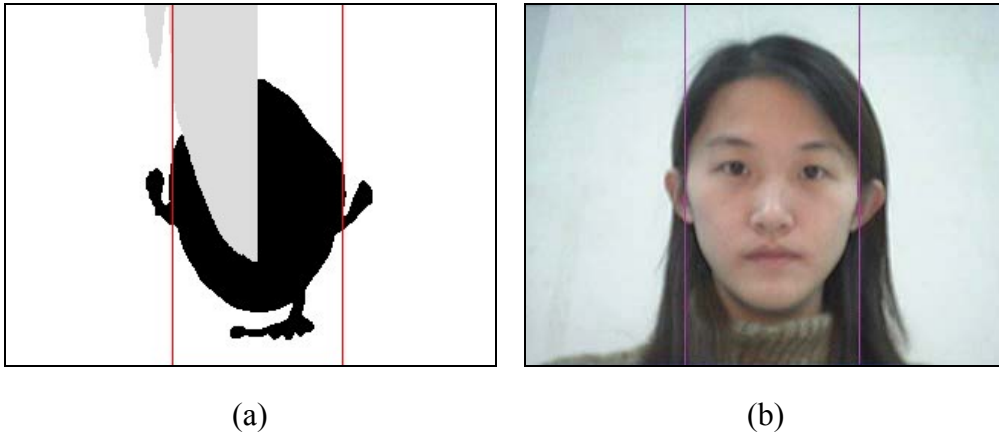


Fig. 3.16 A result of detection of cheek boundaries. (a) Projection information. (b) Cheek boundaries shown on a neutral facial image.

3.6 Experimental Results

Some experimental results of applying the proposed methods for extraction of facial feature regions are shown here. Two different neutral facial images are processed and shown in Fig. 3.17 and Fig. 3.18.

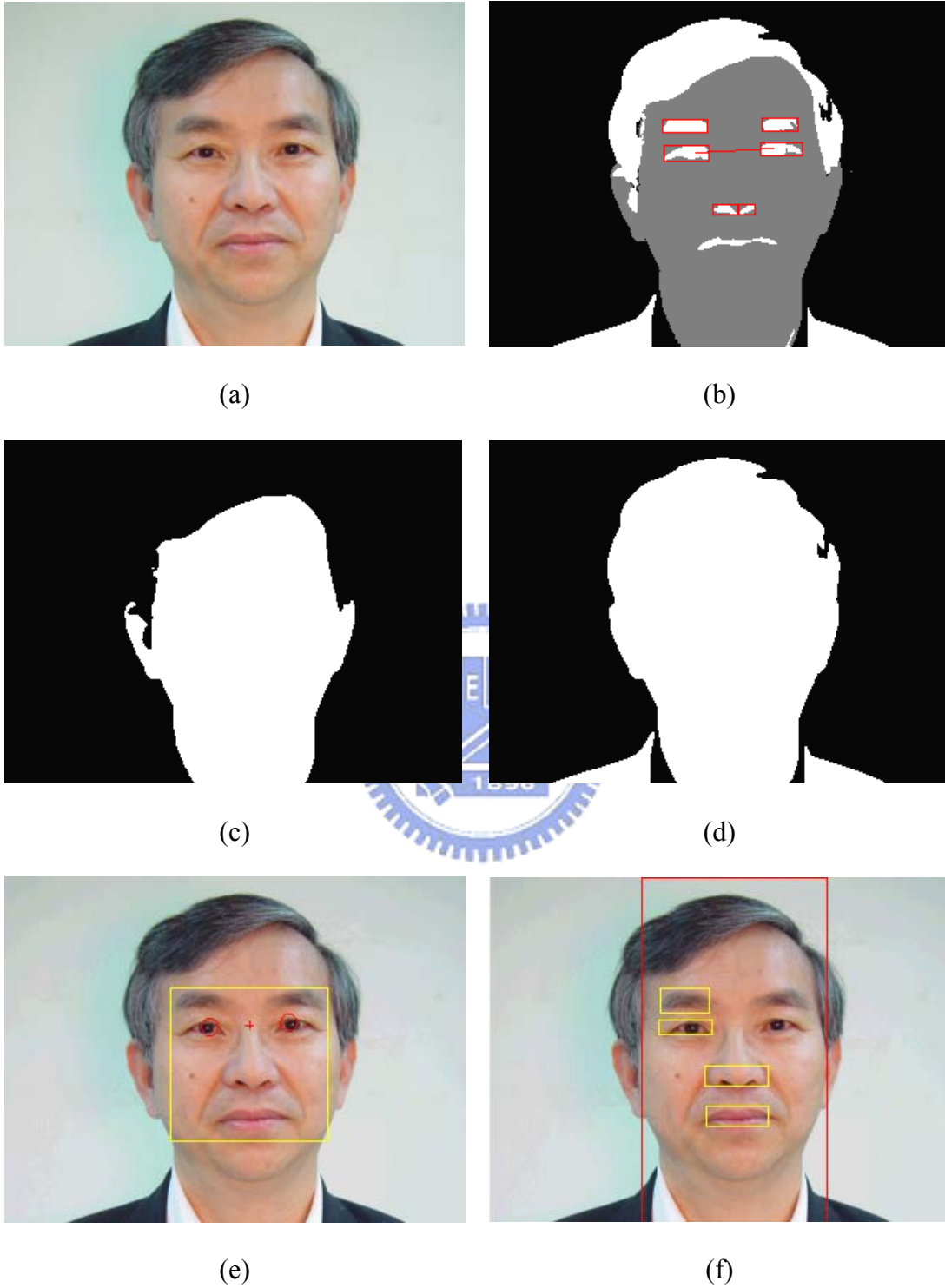


Fig. 3.17 An example of experimental results. (a) A given neutral facial image. (b) A result of optimal eye-pair detection. (c) A result of face regions. (d) A result of hair regions. (e) A result of eyeball detection and $2d \times 2d$ rectangle range. (f) A result of facial feature extraction.

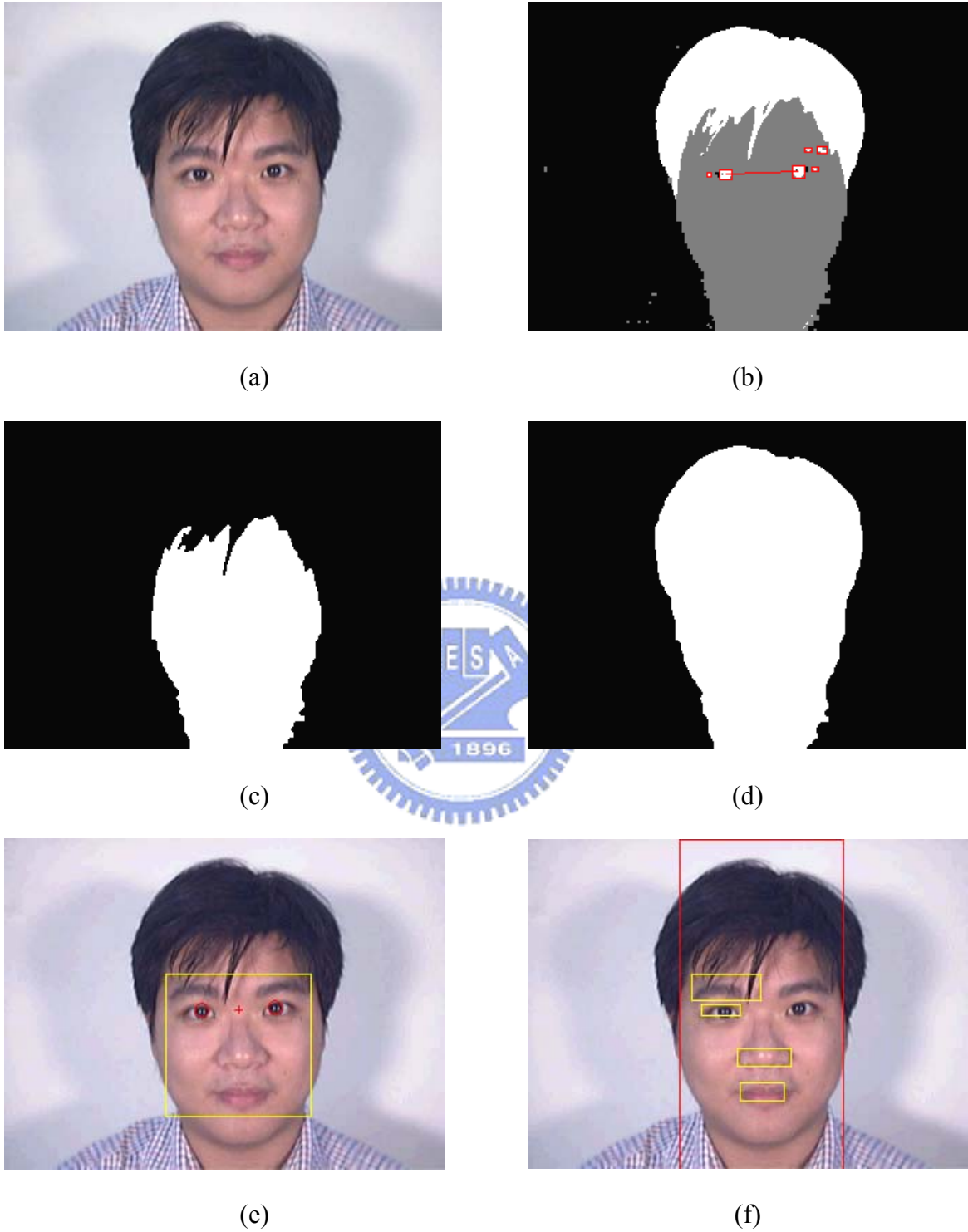


Fig. 3.18 Another example of experimental results. (a) A given neutral facial image. (b) A result of optimal eye-pair detection. (c) A result of face regions. (d) A result of hair regions. (e) A result of eyeball detection and $2d \times 2d$ rectangle range. (e) A result of facial feature extraction.

Chapter 4

Extraction of Facial Feature Points

4.1 Introduction

There are two main reasons for the need of extracting facial feature points. First, the use of facial feature points helps us to draw cartoon faces simply by applying some curve drawing algorithms. Secondly, by reassigning locations to or dragging these facial feature points, we can deform cartoon faces directly.

For the first reason mentioned previously, a face model with facial feature points need be built first. In the MPEG-4 standard, 84 feature points are specified to define facial animation. However, these 84 feature points are not suitable for cartoon face drawing. In this study, a model is designed according to the MPEG-4 standard and two curve drawing algorithms proposed, as described in Section 4.2.

Afterwards, a method is proposed to extract the required facial feature points mentioned previously from the facial feature regions in a given face to draw a personal cartoon face. A more detailed description of the method is given in Section 4.3.

In order to deform cartoon faces more easily, some feature points are assigned to be control points. Besides, some related parameters are extracted for the purposed of face deformation. A more detailed description of the method is given in Section 4.4.

Finally, some experimental results of applying the previously mentioned methods are shown in Section 4.5.

4.2 Face Model Construction

Section 4.2.1 gives a brief review of MPEG-4. TReview of Two Curve Drawing Algorithms are described in Section 4.2.2.

4.2.1 Review of MPEG-4

MPEG-4 is an ISO/IEC standard developed by MPEG (Moving Picture Experts Group), the committee that also developed the Emmy Award winning standards known as MPEG-1 and MPEG-2. MPEG-4 provides the standardized technological elements for the following three fields:

1. digital television;
2. interactive graphics applications (synthetic content);
3. interactive multimedia (World Wide Web, distribution of and access to content)



In Ostermann [12], a part of face animation in MPEG-4 is collected. MPEG-4 specifies a set of facial animation parameters (FAPs), each related to a facial action. In order to provide a reference for defining FAPs, a face model with 84 feature points located in a face is created, as shown in Fig. 4.1.

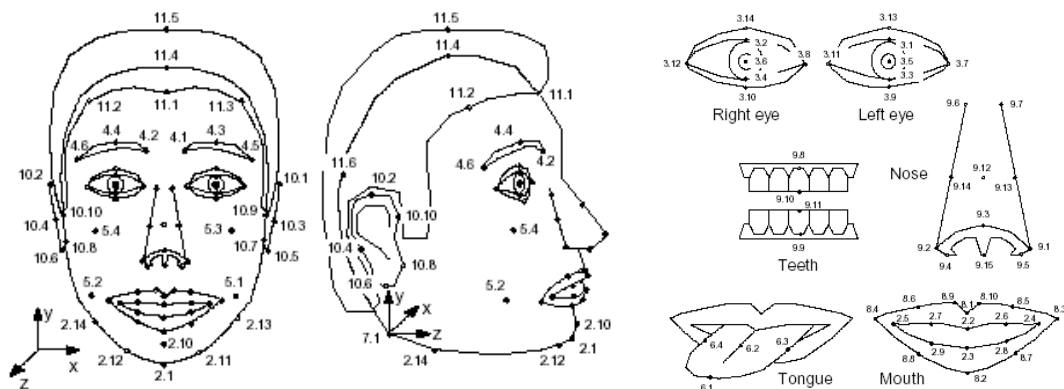


Fig. 4.1 84 feature points in MPEG-4.

Since FAPs represent a set of basic facial actions in which some animation parameters must be defined, facial animation parameter units (FAPUs) are defined as fractions of distances between key facial features, like eye separation ES0, eye-nose separation ENS0, and so on, as shown in Fig. 4.2. The definition of FAPUs allows the FAPs to produce more reasonable animations.



Fig. 4.2 FAPUs in MPEG-4.

However, the feature points and facial animation parameter units are not suitable for cartoon face drawing. For example, a hair contour, maybe of long hair, cannot be represented only by nine feature points including 10.10, 11.2, 11.1, 11.3, 10.9, 10.1, 10.2, 11.5, and 11.4, as shown in Fig. 4.1. Therefore, it is very essential for us to set up an adoptable face model.

4.2.2 Review of Two Curve Drawing Algorithm

In order to set up an adoptable face model mentioned previously, a way to draw the cartoon faces by assigning some feature points must be confirmed first.

Two curve drawing algorithms, including one for curve subdivision and the other for drawing cubic Bezier curves, with different characteristics are designed in the proposed system. They are both supported by the application program SVG, which

will be described in Chapter 7. More detailed illustrations are given in Sections 4.2.2.1 and 4.2.2.2, respectively.

4.2.2.1. Corner-cutting Subdivision

The goal of designing a curve subdivision approximation method is to obtain a smooth curve from a small number of line segments with a set of points. An approach to curve subdivision is corner-cutting subdivision, in which the basic concept is to repeatedly chop off corners of a polygon until reaching a terminating condition, as shown in Fig. 4.3.

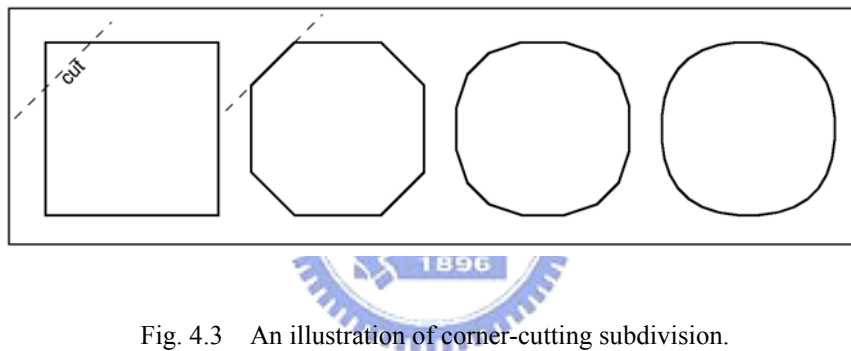


Fig. 4.3 An illustration of corner-cutting subdivision.

The detailed method for corner-cutting is described in the following algorithm, and illustrated in Fig. 4.4.

Algorithm 4.1. *Corner-cutting subdivision algorithm.*

Input: Two line segments $\overline{P_0P_1}$ and $\overline{P_1P_2}$, and a value *depth*.

Output: A curve C_1 .

Steps:

1. If $depth > 0$, compute three mid points P_{01} , P_m , and P_{12} by the following equations.

$$P_{01} = (P_0 + P_1)/2; \quad (4.1)$$

$$P_{12} = (P_1 + P_2)/2; \quad (4.2)$$

$$P_m = (P_{01} + P_{12})/2. \quad (4.3)$$

1.1. Recursively apply Algorithm 4.1 with the following parameters:

$$P_0 = P_0;$$

$$P_1 = P_{01};$$

$$P_2 = P_m;$$

$$width = width - 1.$$

1.2. Recursively apply Algorithm 4.1 with the following parameters:

$$P_0 = P_m;$$

$$P_1 = P_{12};$$

$$P_2 = P_2;$$

$$width = width - 1.$$

2. If $depth \leq 0$, set $C_1 = \overline{P_0P_1} + \overline{P_1P_2}$.

In essence, the above-mentioned curve subdivision method tries to obtain a smooth curve from a set of points. The larger the value of *depth*, the smoother the resulting curve looks like. Additionally, any application language that supports line drawing can be employed to represent the resulting curves of this algorithm.

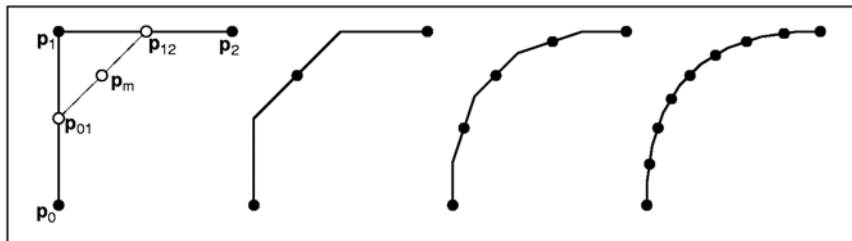


Fig. 4.4 An illustration of corner-cutting algorithm.

4.2.2.2. Cubic Bezier Curve Approximation

Bezier curves are used in computer graphics to produce curves which appear reasonably smooth with a simple polynomial equation in it. It is originally developed by a French automobile engineer Pierre Bézier in the 1970's for CAD/CAM operations. The polynomial equation mentioned previously is illustrated in the following equation:

$$P(t) = \sum_{i=0}^m C_i^m t^i (1-t)^{m-i} P_i, \quad (4.4)$$

where

$$P_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}, \quad C_i^m = \frac{m!}{i!(m-i)!},$$

and $0 \leq t \leq 1$.

In our system, a cubic Bezier curve with $m=4$, which is widely supported in many applications. A cubic Bezier curve can be defined by four points, including an origin endpoint Q_0 , a destination endpoint Q_3 , and two control points (including Q_1 and Q_2). An important property of cubic Bezier curve is that the point R_3 will joint the curve, where R_3 is generated by the following equations, , as illustrated in Fig. 4.5:

$$R_1 = (Q_0+Q_1)/2; \quad (4.5)$$

$$S_2 = (Q_2+Q_3)/2; \quad (4.6)$$

$$R_2 = R_1/2+(Q_1+Q_2)/4; \quad (4.7)$$

$$S_1 = S_2/2+(Q_1+Q_2)/4; \quad (4.8)$$

$$R_3 = (R_2+S_1)/2. \quad (4.9)$$

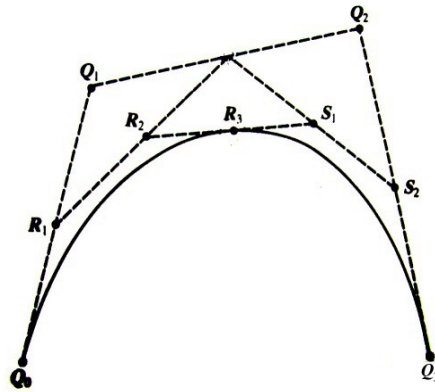


Fig. 4.5 Cubic Bezier curve.

By dragging the points Q_1 and Q_2 , R_3 is also moved. Therefore, by applying this property, a simple curve approximation method is proposed and described in the following.

A. Dragging Q_1 and Q_2 together with the same direction of x or y .

By dragging Q_1 and Q_2 together up, down, left, or right, a new position of R_3 is computed. By recursively measuring the error distance between R_3 and a destination point and the moving the control points, a kind of curve approximation is made. An experimental result of dragging control points downward to implement the curve approximation in which the lowest point is assigned to be the destination point, is illustrated in Fig. 4.6.

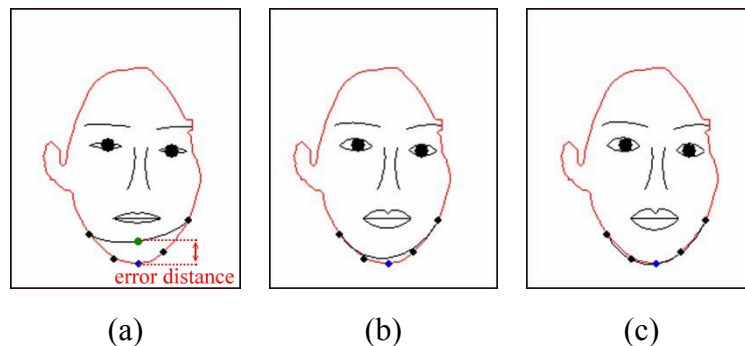


Fig. 4.6 An experimental result of dragging control points with the same direction. (a) Iteration 1 with error distance of 19.75 pixels. (b) Iteration 2 after moving 19 pixels down warding with error distance of 5.75 pixels. (c) Recursively doing the approximation until the error distance smaller than 1 pixel.

B. Dragging Q_1 and Q_2 together with diverse directions of x or y .

By dragging Q_1 and Q_2 to be nearer or further, we can additionally have a sharper or flatter curve. An experimental result is illustrated with changing the jaw shapes in Fig. 4.7.

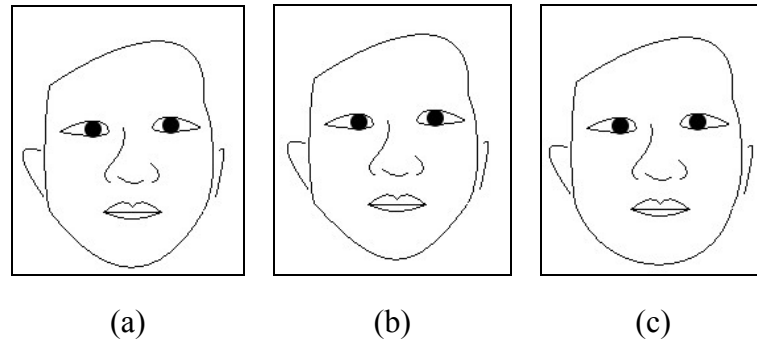


Fig. 4.7 An experimental result of control points with diverse directions. (a) A normal jaw curve. (b) A sharper jaw curve. (c) A flatter jaw curve.

Therefore, the above-mentioned cubic Bezier curve can be used to obtain a smooth curve from four points at a time. Many application languages support the use of skimpily assigning four points to get a Bezier curve. By dragging the control points Q_1 and Q_2 , we can easily change the shapes of Bezier curves.

4.2.3 Proposed Facial Feature Points

By adding or eliminating some feature points of the face model in MPEG-4 according to the two curve drawing algorithms described previously, a new face model with 72 facial feature points is proposed in this study, as illustrated in Fig. 4.8.

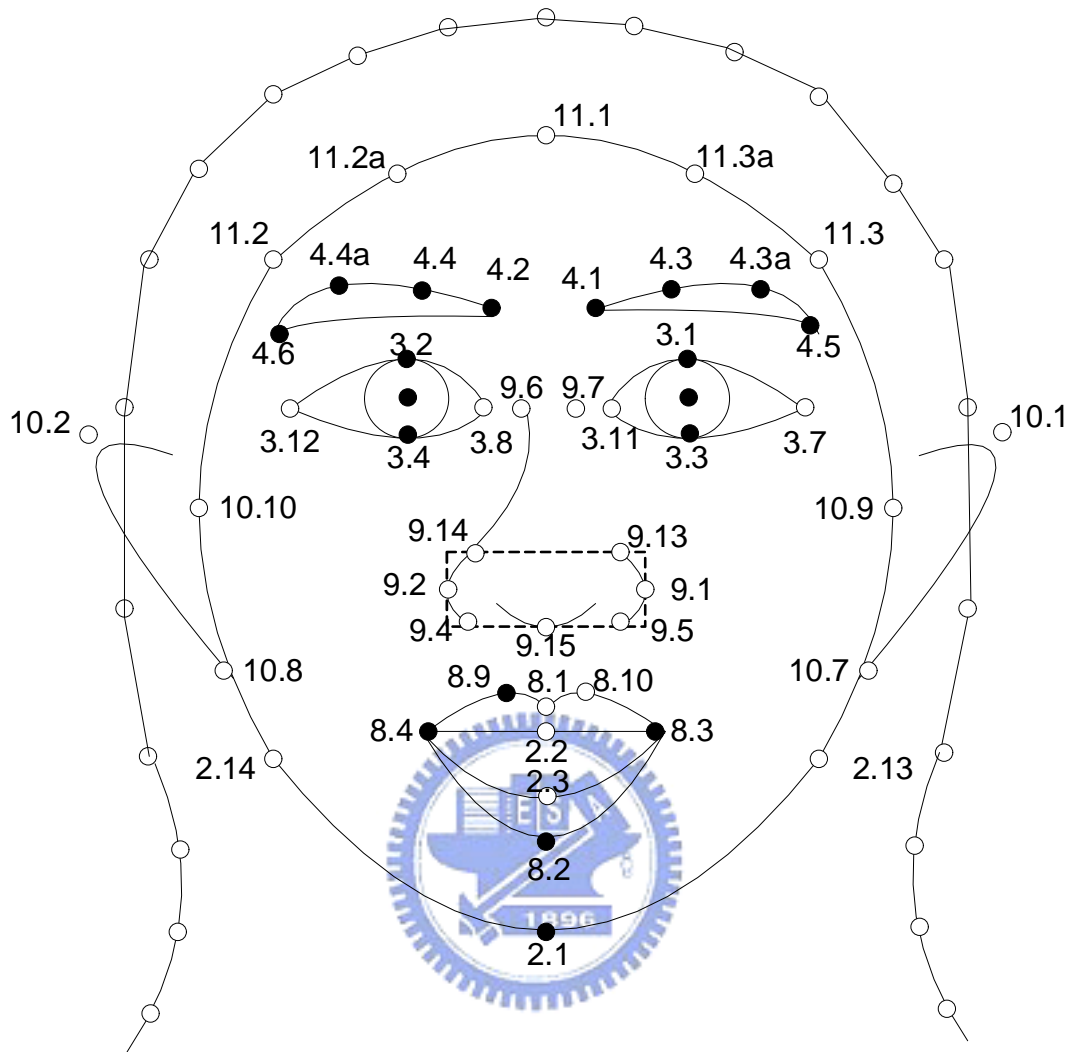


Fig. 4.8 72 feature points in the proposed system.

Utilizing the different properties of the two curve drawing algorithms, we apply a corner-cutting subdivision algorithm with $depth=2$ to draw the hair and eyebrows. On the contrary, we apply the cubic Bezier curve algorithm to draw eyebrows and the remaining facial features.

4.3 Extraction of Facial Feature Points

After setting up the face model with facial feature points, a method for extraction of facial feature points in a given facial image is needed for the purpose of creating a personal cartoon face. In this section, we describe how the facial feature points are extracted in this study from the previously described facial feature regions. By knowing what and how many points of a feature needed to be extracted, a corresponding solution is obtained. In Section 4.3.1, a method for extraction of eyebrows, eyes, nose, and mouth points is proposed and described. Section 4.3.2 describes the method for extraction of face, ear, and hair points.

4.3.1 Extraction of Eyebrows, Eyes, Nose, Mouth, and Jaw Points

In this section, the proposed methods for extraction of feature points based on each facial feature region are described in turn.

A. Extraction of eyebrow points

According to the face model in Section 4.2.3, there are four feature points in each eyebrow region. The detailed extraction method is described in the following algorithm.

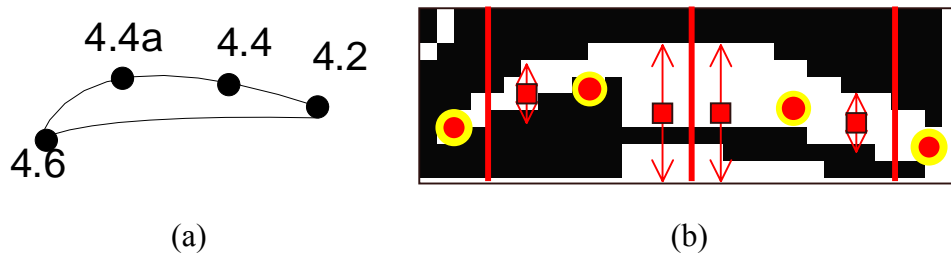


Fig. 4.9 An illustration of extraction of eyebrow points. (a) Left eyebrow points in the proposed face model. (b) Detection of the left eyebrow points in left eyebrow region.

Algorithm 4.2. *Extraction of eyebrow points.*

Input: A left eyebrow region $R_{\text{Left_eyebrow}}$, a value w , and two eyeball regions $\text{Eyeball}_{\text{Left}}$ and $\text{Eyeball}_{\text{Right}}$ detected in Section 3.4.1.

Output: Four left eyebrow points 4.2, 4.4, 4.4a, 4.6, and four right eyebrow points 4.1, 4.3, 4.3a and 4.5.

Steps:

1. Average the leftmost w columns of pixel positions in $R_{\text{Left_eyebrow}}$ as the point of 4.6.
2. Average the rightmost w columns of pixel positions in $R_{\text{Left_eyebrow}}$ as the point of 4.2.
3. Separate the remaining columns of pixels into two parts of regions *Left* and *Right*.
4. For the region *Right*, travel from column to column, and take the average of the centers of all columns as point 4.4.
5. For the region *Left*, travel from column to column, and take the average of the centers of all column as point 4.4a.
6. Compute the right eyebrow points by the left eyebrow points according to the positions of two eyeballs.

B. Extraction of eye points

According to the face model in Section 4.2.3, there are five feature points in each eye region. One of the eye points is the center of the eyeball which was detected in Section 3.4.1. The detailed extraction method of the remaining feature points is described in the following algorithm.

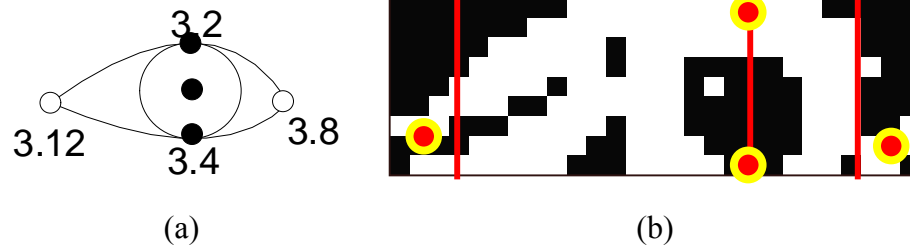


Fig. 4.10 An illustration of extraction of eye points. (a) Left eye points in the proposed face model.
 (b) Detection of the left eye points in left eye region.

Algorithm 4.3. *Extraction of eye points.*

Input: A left eye region $R_{\text{Left_eye}}$, a value w , and two eyeball regions $\text{Eyeball}_{\text{Left}}$ and $\text{Eyeball}_{\text{Right}}$ detected in Section 3.4.1.

Output: Four left eye points 3.8, 3.12, 3.2, 3.4, four right eye points 3.7, 3.11, 3.1, and 3.3.

Steps:

1. Average the leftmost w columns of pixel positions in $R_{\text{Left_eye}}$ as the point of 3.12.
2. Average the rightmost w columns of pixel positions in $R_{\text{Left_eye}}$ as the point of 3.8.
3. Set point 3.2 according to the definition of the neutral face in the following way:

$$\text{Point } 3.2 = (\text{Eyeball}_{\text{Left}}.x, \text{Eyeball}_{\text{Left}}.y - \text{Eyeball}_{\text{Left}}.height / 2).$$

4. Set point 3.4 according to the definition of the neutral face in the following way:

$$\text{Point } 3.4 = (\text{Eyeball}_{\text{Left}}.x, \text{Eyeball}_{\text{Left}}.y + \text{Eyeball}_{\text{Left}}.height / 2).$$

5. Compute the right eye points by the left eye points according to the positions of two eyeballs.

C. Extraction of nose points

According to the face model in Section 4.2.3, there are nine feature points in the nose region. Since the edge information of the nose region is not always adoptable for pixel-by-pixel detection, here, we only use some given information to get the desired feature points. The detailed method is described in the following algorithm.

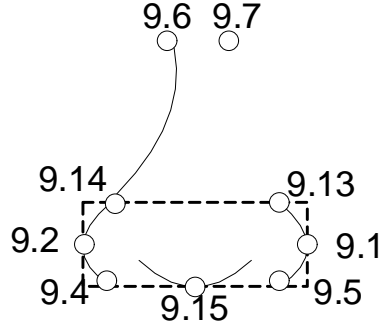


Fig. 4.11 Left eye points in the proposed face model.

Algorithm 4.4. Extraction of nose points.

Input: A nose region R_{Nose} , two eye points (including 3.8 and 3.11), a vertical cut L_{Nose} detected in Section 3.4.3, a position of $EyeMid$, and two eyeball regions $Eyeball_{\text{Left}}$ and $Eyeball_{\text{Right}}$ detected in Section 3.4.1.

Output: Nine nose points 9.6, 9.7, 9.14, 9.13, 9.2, 9.1, 9.4, 9.5, and 9.15.

Steps:

1. Let $R_{\text{Nose}} = [(x_1, y_1), (x_2, y_2)]$.
2. Let L_{Nose} denote a line described by $x=x_N$.
3. Set point 9.6 and 9.7 in the following way:
 Point 9.6 = $(3.8.x + (3.11.x - 3.8.x)/3, EyeMid.y)$;
 Point 9.7 = $(3.11.x - (3.11.x - 3.8.x)/3, EyeMid.y)$.
4. Set point 9.14 and 9.13 in the following way:
 Point 9.14 = $(x_1 + (x_2 - x_1)/8, y_1)$;

$$\text{Point 9.13} = (x_2 - (x_2 - x_1)/8, y_1).$$

5. Set point 9.2 and 9.1 in the following way:

$$\text{Point 9.2} = (x_1, (y_1 + 3 \times y_2)/4);$$

$$\text{Point 9.1} = (x_2, (y_1 + 3 \times y_2)/4).$$

6. Set point 9.4 and 9.5 in the following way:

$$\text{Point 9.4} = (x_1 + (x_2 - x_1)/8, y_2);$$

$$\text{Point 9.5} = (x_2 - (x_2 - x_1)/8, y_2).$$

7. Set point 9.15 in the following way:

$$\text{Point 9.15} = (x_N, y_2).$$

D. Extraction of mouth points

According to the face model in Section 4.2.3, there are eight feature points in the mouth region. Since the mouth shape itself has a symmetry property, we will only detect the left-side feature points and then compute the remaining feature points. The detailed method is described in the following algorithm.



Fig. 4.12 An illustration of extraction of mouth points. (a) Mouth points in the proposed face model.

(b) Detection of the mouth points in mouth region.

Algorithm 4.5. *Extraction of mouth points.*

Input: A mouth region R_{Mouth} , a value w , and a vertical cut L_{Mouth} detected in Section 3.4.3

Output: Eight mouth points 8.1, 8.9, 8.10, 8.4, 8.3, 2.2, 2.3, and 8.2.

Steps:

1. Let R_{Mouth} denotes a region of $[(x_1, y_1), (x_2, y_2)]$.
2. Let L_{Mouth} denote a line described by $x=x_M$.
3. Average the leftmost w columns of pixel positions in R_{Mouth} as the point of 8.4.
4. Average the topmost w rows of pixel positions in the half part of R_{Mouth} as the point of 8.9.
5. Average the topmost $(8.4.y - x_1)$ pixel position along the vertical cut L_{Mouth} as the point of 8.1.
6. Set point 2.2, 2.3, 8.2 in the following way:
Point 2.2 = $(x_M, 8.4.y)$;
Point 2.3 = $(x_M, 8.4.y)$;
Point 8.2 = (x_M, y_2) .
7. Compute points 8.10 and 8.3 according to the central vertical cut L_{Mouth} .

4.3.2 Extraction of Face, Ear, and Hair Points

In this section, the proposed method for extraction of face, ear, and hair points is described. The main idea to extract these feature points is to choose a start point and searching. When the start point moves to a position that satisfies a predefined rule, the search is stopped and the position is record as one feature point. This method is sensitive to noise, and so we need to refine the face region and hair region, as described in Chapter 3.

A. Extraction of face points (including jaw points), and ear points

In Chapter 3, we already knew the face region and ear region. Here we only describe the proposed method for extraction of the major face points. An illustration is shown in Fig. 4.13. Some basic rules to extract face points (including points 2.1, 2.14, 2.13, 11.2, 11.1, and 11.3) and nose points (including points 10.2, and 10.10) are listed as follows:

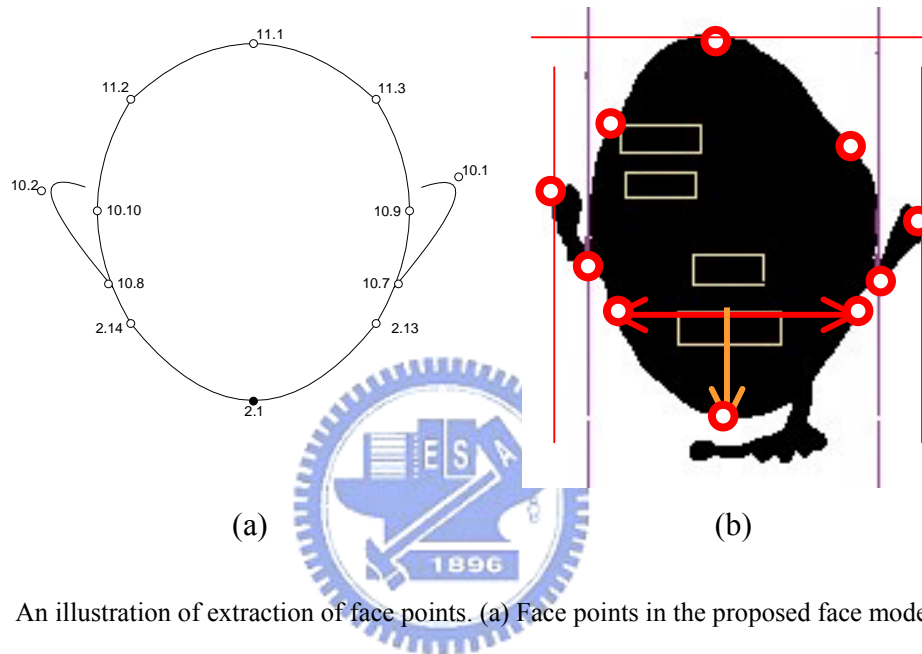


Fig. 4.13 An illustration of extraction of face points. (a) Face points in the proposed face model.

(b) Detection of the face points in face region.

1. Start from the mouth point 8.2, search downward until reaching a position which does not belong to the face region F_{final} , and take it as the jaw point 2.1. If we cannot detect point 2.1, we speculate a position for it in the following way:

$$\text{Point } 2.1 = (8.2.x, 8.2.y + 0.7 \times d).$$

2. Start from the mouth point 8.9, search to the left and right simultaneously, until reaching positions which do not belong to the face region F_{final} as another two jaw points 2.14, 2.13. If the cheek boundary is reached, then backtrack to the position of point 8.9, move downward 1 pixel, and start search again.

3. Start from position $(2.14.x, MidEye.y+d/2)$, search upward, until reaching a position which does not belong to the face region F_{final} , and take it as the face point 11.2.
4. Start from position $(2.13.x, MidEye.y+d/2)$, search like the way for point 11.2 to get a face point 11.3.
5. Average the positions of the topmost several rows in the face region F_{final} to get face point 11.1.
6. Finally, ear points can be detected by averaging the outmost several rows to get ear points 11.2 and 11.1.

B. Extraction of hair points

An illustration is shown in Fig. 4.14. Some basic rules to extract hair points are listed as follows:

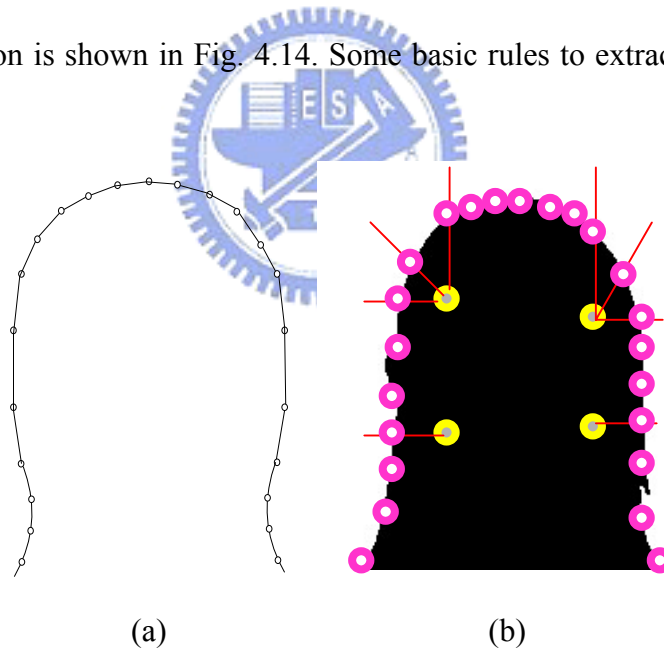


Fig. 4.14 An illustration of extraction of hair points. (a) Hair points in the proposed face model.

(b) Detection of the hair points in hair region.

1. Us the points 2.14, 2.13, 11.2, and 11.3 as basic points to divide the hair contour with several segments, as shown in Fig. 4.14(b).
2. Us one of the points 2.14, 2.13, 11.2, or 11.3 as a start point, search outward,

until reaching a position which does not belong to the hair region H_{final} , and take it as one of the hair points.

4.4 Cartoon Face Deformation

In order to animate cartoon faces more easily, some feature points are assigned to be control points. The major moving elements are eyes, the mouth, and the jaw. To move them in a more reasonable way, some animation parameter units need to be specified, as described in Section 4.4.1. A way to animate cartoon faces in a simpler way is described in Section 4.4.2.

4.4.1 Proposed Facial Animation Parameter Units

Some animation parameter units used in the proposed system is illustrated in Fig. 4.15. Each can be estimated by computing the Euclidean distance between two feature points.

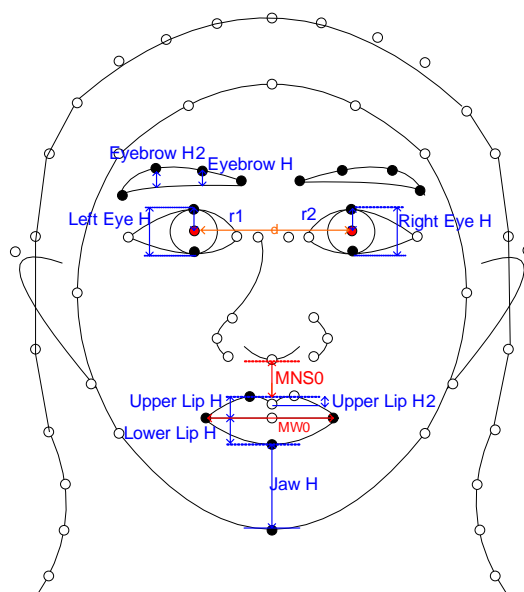


Fig. 4.15 Facial animation parameter units in the proposed system.

4.4.2 Assignment of Control Points

Eyes and the mouth are basic animation elements in this study. An easier way to control their actions can simplify the animation process.

For the action of eyes, we can simply change the position of the eyeball to make an eye movement effect. Additionally, by changing the value of *LeftEyeH* and *RightEyeH*, we can easily create an eye blinking effect. Points and parameters related to these effects are illustrated in the following figure.

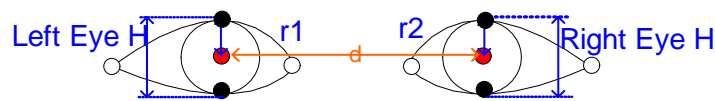


Fig. 4.16 An illustration of eye deformations.

Next, in order to control the movement of the lips more easily, a method is also proposed in this study. By reassigning the locations of mouth points 8.9, 8.4, 8.2, and 8.3, the other mouth points will be computed automatically. Additionally, by using the facial animation parameters *MNS0* and *MW0*, we can automatically simulate thick lips or thin lips for use during speaking. The jaw point 2.1 is automatically computed without reassigning its location. Moving mouth 8.9 will make the jaw point 2.1 moving simultaneously with the parameter *JawH*, where point 2.1 = $(8.2.x, 8.2.y + \text{JawH})$. Points and parameters related to these effects are illustrated in the following figure.

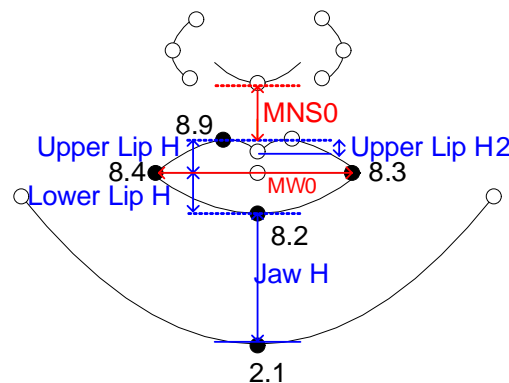


Fig. 4.17 An illustration of mouth deformations.

4.5 Experimental Results

Some experimental results of applying the proposed methods for extraction of facial feature points are shown here. Two different neutral facial images are processed and shown in the following.

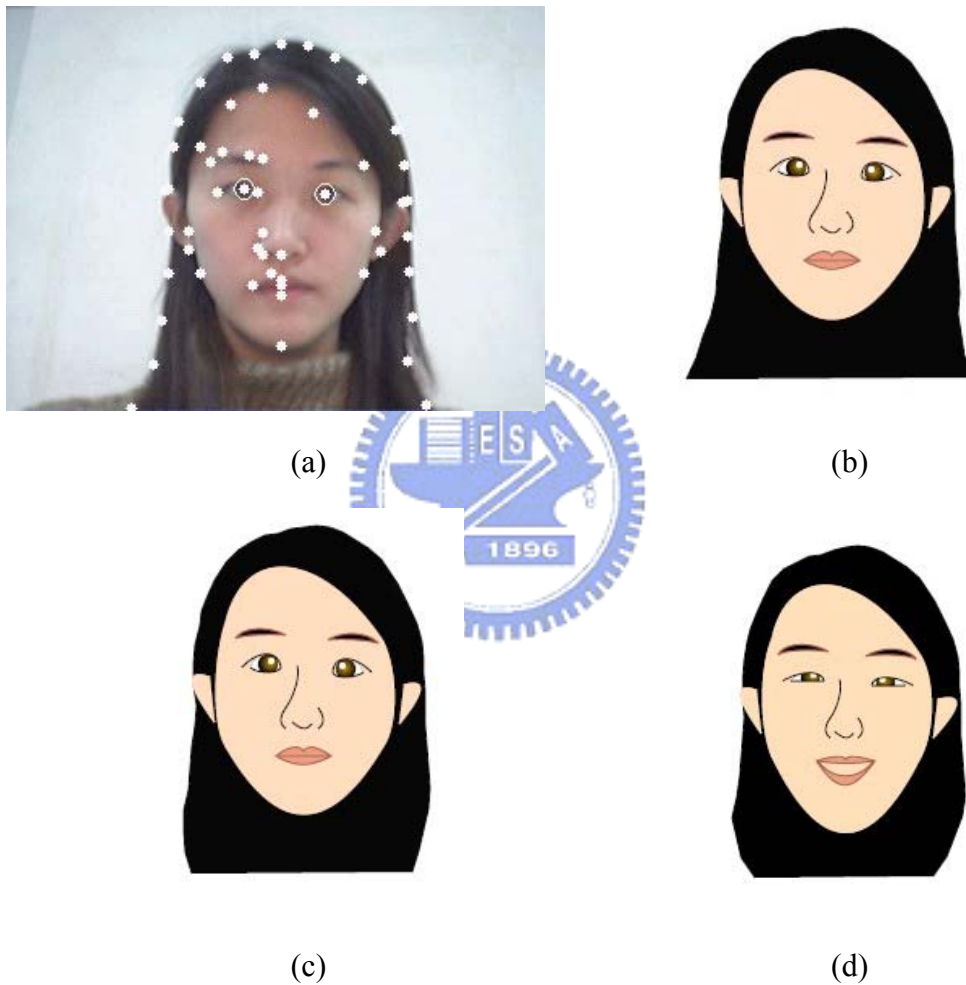
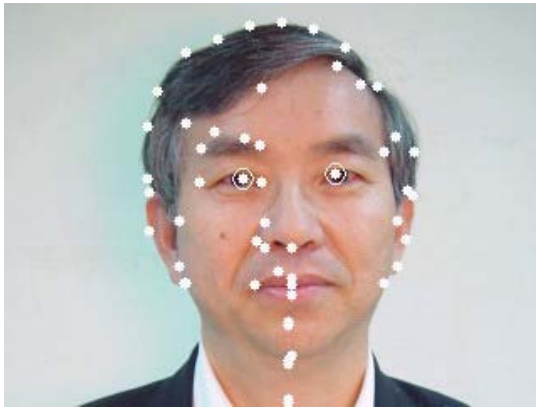
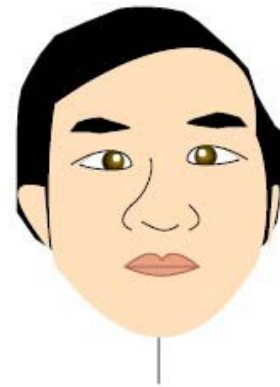


Fig. 4.18 An example of experimental results. (a) A result of detection of facial feature points. (b) A result of neutral cartoon face rendered by SVG. (c) A result of a cartoon face with relocating some facial points. (d) A result of a cartoon face with relocating the control points of eyes and mouth.



(a)



(b)



(c)

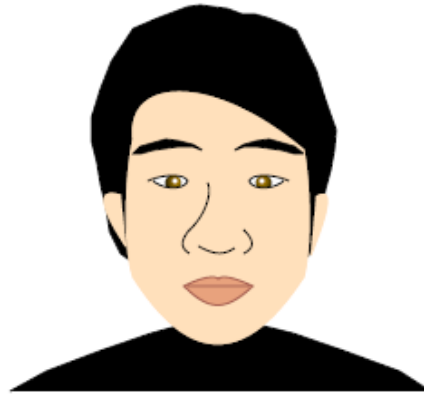


(d)

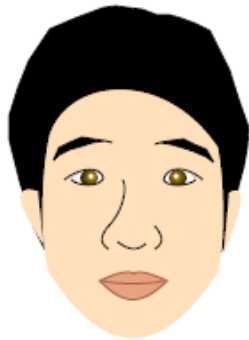
Fig. 4.19 The second example of experimental results. (a) A result of detection of facial feature points. (b) A result of neutral cartoon face rendered by SVG. (c) A result of a cartoon face with relocating some facial points. (d) A result of a cartoon face with relocating the control points of eyes and mouth.



(a)



(b)



(c)



(d)

Fig. 4.20 The third example of experimental results. (a) A result of detection of facial feature points. (b) A result of neutral cartoon face rendered by SVG. (c) A result of a cartoon face with relocating some facial points. (d) A result of a cartoon face with relocating the control points of eyeballs (shifting up) and mouth.

Chapter 5

Talking Cartoon Face Generation from Single Images

5.1 Introduction

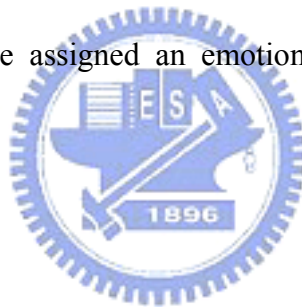
Up to now, we have known how to create a personal cartoon face from a given neutral facial image, how to change its looks, and how to animate it using the techniques described in the previous chapters. However, we do not want to create an animated cartoon face by dealing with it frame by frame. In this study, two ways are proposed to help users to create a talking cartoon face automatically for general uses. The first is to create a talking cartoon face from a single facial image, but animate it by using a speech file and a script file. That is, users, for example, can take pictures to make cartoon faces whenever they want their personal cartoon faces to speak on behalf of themselves.

On the contrary, if a user wants to create an animated cartoon face with fewer operations, the second way may be employed to attain the goal by simply using a video clip of the user's talking face as input. Then the generated talking cartoon face will act like the user's face in the video clip. Furthermore, cartoon faces are created with less data, and so people can easily share them through the Internet, cellular phones, etc, with their friends.

In this chapter, the first way mentioned previously is described, and the second way is described in the next chapter.

In order to animate the cartoon faces from a speech file and a script file, we have to know the relationships among them. By applying a speech analyzer, time information of the syllables of the speech file can be extracted in this study. However, knowing when to speak a word is still not enough. We must know “how” to speak. Nine *basic* mouth shapes are proposed to translate syllables into combinations of several basic mouth shapes. By assigning basic mouth shapes into proper key frames, the cartoon face can be animated.

In Section 5.2, a definition of basic mouth shapes is described. In order to know the relationship between the mouth shapes, an analysis of time intervals between mouth shapes of syllables is given in Section 5.3. Next, the stage of talking cartoon face generation by synthesizing moving lips is described in Section 5.4. Additionally, a talking cartoon face can be assigned an emotional expression in this study, as described in Section 5.5.



5.2 Definition of Basic Mouth Shapes

Chinese words are monosyllables with 21 initials, 38 finals, and 5 tones. There are 1345 syllables for Mandarin speaking. If the differences in tones are ignored, the 1345 syllables can be reduced to 411 syllables. In Yeh [16], the 21 kinds of initials were reduced to 7 according to the manners of articulation. After a classification of the initials, the 411 syllables can even be reduced to 167 postures.

However, the above mentioned classification techniques are based on “real” faces. In the case of cartoon faces, especially when the concern is about the mouth shapes only, the number of postures can be reduced further, as done in this study.

5.2.1 Phonetic Transcription

There exist many languages in our world. In order to read an unknown word, people always transcript it into some basic phonetic symbols. For Mandarin speaking, some basic phonetic transcriptions, such as .the Chinese phonetic transcription and the Taiwan Tongyoung Romanization, have already existed. Taking the Taiwan Tongyoung Romanization as an example, we show a transcription of the Mandarin phonetics into a set of English alphabets in Table 5.1.

Table 5.1 An illustration of the basic elements of Taiwan Tongyoung Romanization.

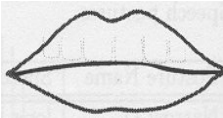

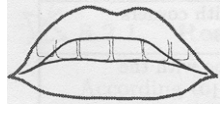

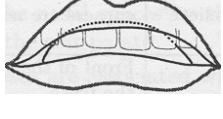

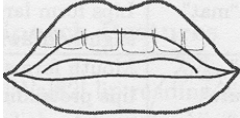

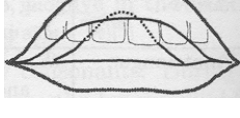
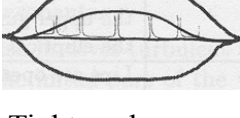
| | | | | | | | |
|----|----|----|----|----|-----|----|-----|
| ㄅ | ㄆ | ㄇ | ㄈ | ㄉ | ㄊ | ㄋ | ㄌ |
| b | p | m | f | d | t | n | l |
| ㄍ | ㄎ | ㄏ | ㄐ | ㄑ | ㄒ | | |
| g | k | h | ji | ci | si | | |
| ㄐ | ㄑ | ㄒ | ㄓ | ㄔ | ㄕ | ㄌ | |
| jr | ch | sh | r | z | c | s | |
| ㄚ | ㄛ | ㄜ | ㄝ | ㄞ | ㄟ | ㄠ | ㄡ |
| a | o | e | i | u | u | e | er |
| ㄟ | ㄠ | ㄡ | ㄢ | ㄣ | ㄤ | ㄥ | ㄨ |
| ai | ei | ao | ou | an | ang | en | eng |

We can see from Table 5.1 that the Mandarin phonetics are composed of some basic English alphabets, and that each English alphabet has its own mouth shapes. A concept worth notice here is that a Mandarin syllable which consists of phonemes is a combination of some basic mouth shapes.

5.2.2 Extraction of Basic Mouth Shapes

In Yeh [16], the 21 kinds of initials, as mentioned previously, were reduced to 7 according to the manners of articulation, as illustrated in Table 5.2:

Table 5.2 Classification of initials according to the manner of articulation proposed in Yeh [16].

| Pace of Articulation | | Members of the Classes |
|---|--|------------------------|
| Upper Obstruction | Lower Obstruction | |
|  <p><i>Lip shut</i></p> |  <p><i>Relaxed narrow</i></p> | フ、フ、フ |
|  <p>Protrusion</p> |  <p>Relaxed narrow</p> | ク |
|  <p>Tongue to gum</p> |  <p>Medium oval</p> | フ、ク、ク、ク |
|  <p>Medium oval</p> | | ク、ク、ク |
|  <p>Front tongue</p> | | ク、ク、ク |
|  <p>Tip to gum</p> | | ク、ク、ク、ク |
|  <p>Tightened narrow</p> | | ク、ク、ク |

Because talking cartoon face generation by the proposed system only concern about the mouth shapes of the pronunciation, the above mentioned 7 kinds of Mandarin initials can be reduced further to 3 basic initial mouth shapes, as shown in Table 5.3.

Table 5.3 Three basic mouth shapes of Mandarin initials.

| Mouth Shape Symbols | Members of the classes |
|---------------------|-------------------------------------|
| <i>m</i> | ㄇ、ㄇ、ㄇ |
| <i>f</i> | ㄈ |
| <i>h'</i> | ㄏ、ㄏ、ㄏ、ㄏ、ㄏ、ㄏ、ㄏ、ㄏ、ㄏ、ㄏ、 ㄏ、ㄏ、ㄏ、ㄏ、ㄏ、ㄏ |

Similarity, the Mandarin finals can be reduced to a set of combinations with 7 basic mouth shapes based on the Taiwan Tongyoung Romanization, as shown in Table 5.4.

Table 5.4 A set of combinations with 7 basic mouth shapes of Mandarin finals.

| Mouth Shape Symbols | Members of the classes | Combinations of Mouth Shapes | Members of the classes |
|---------------------|------------------------|------------------------------|------------------------|
| <i>a</i> | ㄚ | <i>au</i> | ㄨ |
| <i>e</i> | ㄝ | <i>ou</i> | ㄨ |
| <i>i</i> | ㄟ | <i>en</i> | ㄨ |
| <i>o</i> | ㄛ | <i>an</i> | ㄨ |
| <i>u</i> | ㄨ、ㄨ | <i>ei</i> | ㄨ |
| <i>n</i> | ㄨ | <i>ai</i> | ㄨ |
| <i>h</i> | ㄨ、ㄨ、ㄨ | | |

Any given syllable which consists of phonemes can be translated into a combination of basic mouth shape symbols. For example, phoneme “ \times ” can be translated into “ u ”, phoneme “ $\text{ㄝ$ ” into “ au ”, and a syllable “ ㄟ ” into “ iau ”, etc.

It is noticed that two “ h ’s” exist in both of the classes of Mandarin initials and Mandarin finals. Both h' and h represent the same open mouth shape; however, h' is eliminated in this study if there is a symbol (including a , i , u , e , o) behind it. For example, the syllable “ ㄏㄨ ” is translated into “ $h'u$ ”, where there is an u right behind h' , and the final transcription of “ ㄏㄨ ” will be automatically corrected to be “ u .”

The reason for the above mentioned rule is that for Mandarin speaking, the pre-posture of the open mouth shape is always missed. Lin and Tsai [1] only checked the initial of the syllable if it is one of the phonemes “ ㄅ ”, “ ㄆ ”, “ ㄇ ”, “ ㄏ ” by ignoring the open mouth shapes of the remaining phonemes of the Mandarin initials.

5.2.3 Basic Mouth Shapes

After classifying the Mandarin initials and the Mandarin finals into 9 basic mouth shapes, the shape of each need be defined. An experiment was carried out to define the basic mouth shapes for uses of cartoon faces. Additionally, we can easily define the mouth shape using the concept of control points, which is mentioned previously in Chapter 4, as shown in Fig. 5.1.

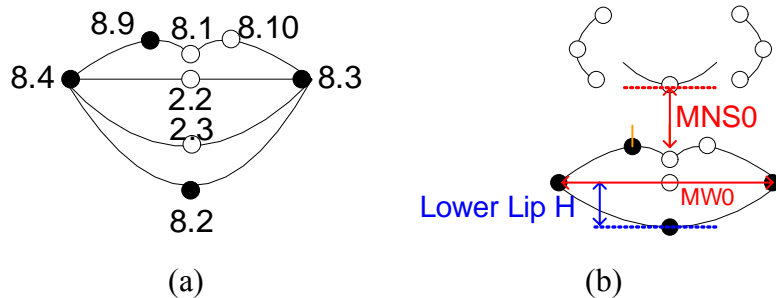


Fig. 5.1 Basic components for definition of basic mouth shapes. (a) Control points (solid circle) of mouth. (b) Facial animation parameter units of mouth and nose.

Let each basic mouth shape be defined by a set of {8.9, 8.2, 8.4, 8.3}. The relevant definitions are as follows.

A. Basic mouth shapes of Mandarin initials.

1. Basic mouth shape m : m is the initial mouth shape in the neutral state:

$$m = \{8.9_{\text{Neutral}}, 8.2_{\text{Neutral}}, 8.4_{\text{Neutral}}, 8.3_{\text{Neutral}}\}.$$

2. Basic mouth shape f : A part of the lower lip is hidden by the upper lip:

$$f = \{8.9_{\text{Neutral}}, up, 8.4_{\text{Neutral}}, 8.3_{\text{Neutral}}\},$$

$$\text{where } up = (8.9_{\text{Neutral}.x}, 8.9_{\text{Neutral}.y} - 0.5 \times \text{LowerLipH}).$$

3. Basic mouth shape h' and h :

$$h'/h = \{up, dn, lf, rt\},$$

$$\text{where } up = (8.9_{\text{Neutral}.x} - 0.3 \times \text{MNSO}, 8.9_{\text{Neutral}.y});$$

$$dn = (8.2_{\text{Neutral}.x}, 8.2_{\text{Neutral}.y} + 0.3 \times \text{MNSO});$$

$$lf = (8.4_{\text{Neutral}.x}, 8.4_{\text{Neutral}.y} - 1);$$

$$rt = (8.3_{\text{Neutral}.x}, 8.3_{\text{Neutral}.y} - 1).$$

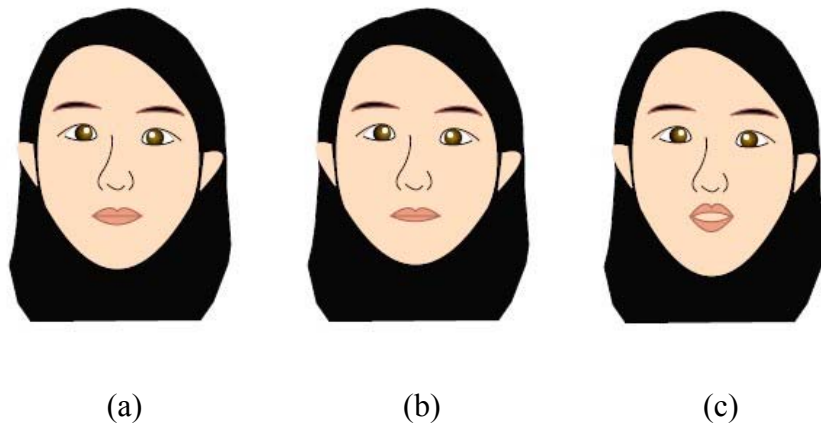


Fig. 5.2 Basic mouth shapes of Mandarin initials rendered by SVG. (a) Basic mouth shape m . (b) Basic mouth shape f . (c) Basic mouth shape h .

B. Basic mouth shape of Mandarin finals.

1. Basic mouth shape *a*:

$$a = \{up, dn, lf, rt\},$$

$$\text{where } up = (8.9_{\text{Netural}.x}, 8.9_{\text{Netural}.y} - 0.3 \times MNS0);$$

$$dn = (8.2_{\text{Netural}.x}, 8.2_{\text{Netural}.y} + 0.8 \times MNS0);$$

$$lf = (8.4_{\text{Netural}.x} + MW0/8, 8.4_{\text{Netural}.y});$$

$$rt = (8.3_{\text{Netural}.x} - MW0/8, 8.3_{\text{Netural}.y}).$$

2. Basic mouth shape *i*:

$$i = \{8.9, dn, lf, rt\},$$

$$\text{where } dn = (8.2_{\text{Netural}.x}, 8.2_{\text{Netural}.y} + MNS0/6);$$

$$lf = (8.4_{\text{Netural}.x} - MW0/12, 8.4_{\text{Netural}.y} - 3);$$

$$rt = (8.3_{\text{Netural}.x} + MW0/12, 8.3_{\text{Netural}.y} - 3).$$

3. Basic mouth shape *u*:

$$u = \{up, dn, lf, rt\},$$

$$\text{where } up = (8.9_{\text{Netural}.x}, 8.9_{\text{Netural}.y} - 0.3 \times MNS0);$$

$$dn = (8.2_{\text{Netural}.x}, 8.2_{\text{Netural}.y} + 0.3 \times MNS0);$$

$$lf = (8.4_{\text{Netural}.x}, 8.4_{\text{Netural}.y} - 1);$$

$$rt = (8.3_{\text{Netural}.x}, 8.3_{\text{Netural}.y} - 1).$$

4. Basic mouth shape *e* :

$$e = \{8.9, dn, lf, rt\},$$

$$\text{where } dn = (8.2_{\text{Netural}.x}, 8.2_{\text{Netural}.y} + MNS0/2);$$

$$lf = (8.4_{\text{Netural}.x} - MW0/12, 8.4_{\text{Netural}.y} - 3);$$

$$rt = (8.3_{\text{Netural}.x} + MW0/12, 8.3_{\text{Netural}.y} - 3).$$

5. Basic mouth shape *o* :

$$o = \{up, dn, lf, rt\},$$

$$\text{where } up = (8.9_{\text{Netural}.x}, 8.9_{\text{Netural}.y} - 0.3 \times MNS0);$$

$$dn = (8.2_{\text{Netural}.x}, 8.2_{\text{Netural}.y} + 0.5 \times MNS0);$$

$$lf = (8.4_{\text{Netural}.x} + 0.2 \times MW0, (up.y + dn.y) / 2);$$

$$rt = (8.3_{\text{Netural}.x} - 0.2 \times MW0, lf.y).$$

6. Basic mouth shape n :

$$n = \{up, dn, lf, rt\},$$

where $up = (8.9_{\text{Netural}.x}, 8.9_{\text{Netural}.y} - 0.25 \times MNS0);$

$$dn = (8.2_{\text{Netural}.x}, 8.2_{\text{Netural}.y} + 0.5 \times MNS0);$$

lf = equals to the last former mouth shape;

rt = equals to the last former mouth shape.

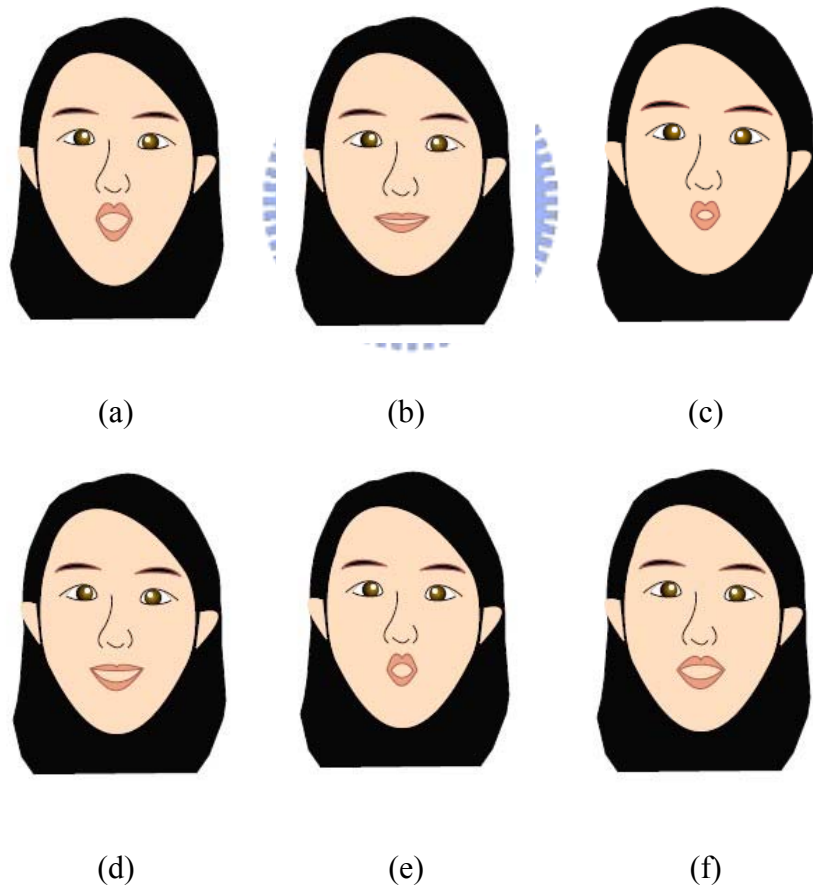


Fig. 5.3 Basic mouth shapes of Mandarin finals rendered by SVG. (a) Basic mouth shape a . (b) Basic mouth shape i . (c) Basic mouth shape u . (d) Basic mouth shape e . (e) Basic mouth shape o . (f) Basic mouth shape n .

5.3 Analysis of Time Intervals between Mouth Shapes of Syllables

A syllable is combined with four basic mouth shapes at most in this study. We have to know when to show each mouth shape during the pronunciations of syllables.

Since the mouth shapes are extracted according to the Taiwan Tongyoung Romanization, each mouth shape of a syllable can be roughly recognized by general listening. Some experiments were conducted to determine the time intervals between mouth shapes. We choose the Mandarin initials and Mandarin finals as the samples for the experiments. After recording an audio with several continuous syllables, we select a part of a syllable, and then try to find the time interval between two pronunciations of the basic mouths by carefully listening.

For example, a syllable “ㄟ” is translated into “ei” by the previously mentioned method. We suppose that there is a start time for the pronunciation of the basic mouth shape “i” and try to recognize it, as shown in Fig. 5.4.

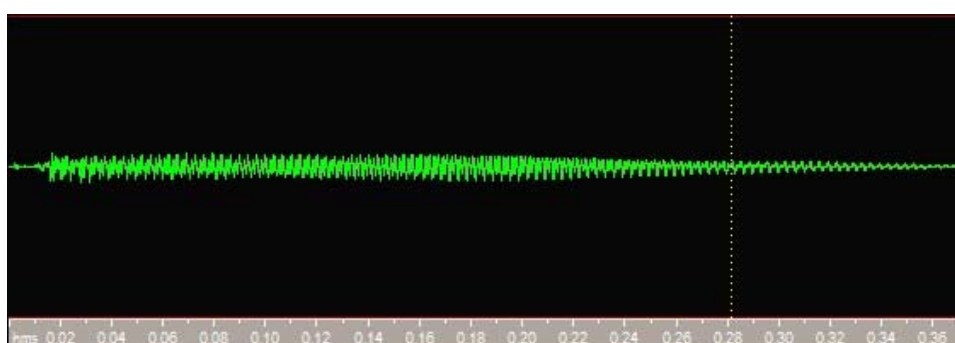


Fig. 5.4 The waveform of the syllable “ㄟ” and the start time of the second phoneme.

Repeatedly applying the work for all the Mandarin initials and all the Mandarin finals, we get some experimental results of syllables with one to three basic mouth shapes. Therefore, by combining the Mandarin initials and the Mandarin finals, we

can get the syllables with four basic mouth shapes.

Some experimental results are shown in the following.

A. A syllable with two basic mouth shapes.

There are 44 Mandarin finals with 2 basic mouths. By repeatedly doing three times, the experimental results of gathering the start times of the first time intervals of the pronunciations of the basic mouth shapes are illustrated in Fig. 5.5.

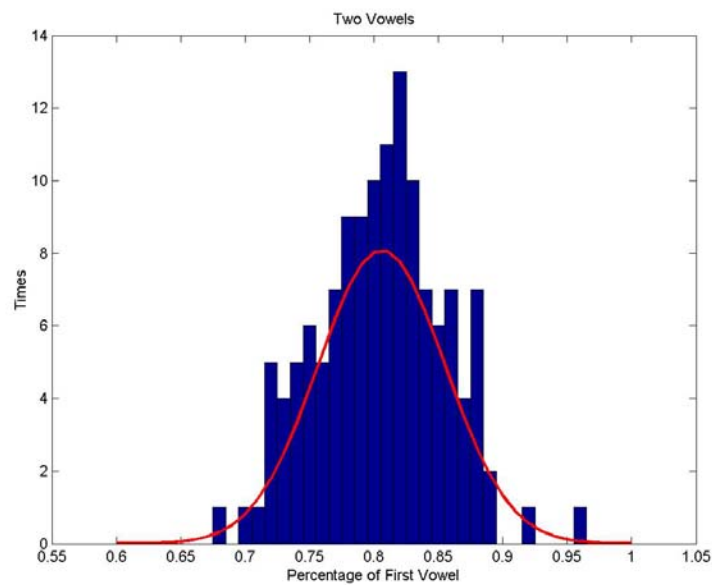


Fig. 5.5 A statistic result of the start time of the second basic mouth shape of the syllables with 2 basic mouth shapes.

A normal distribution with $mean = 0.8095$ and $variance = 0.0494$ normalized by the total audio time was obtained. Therefore, we assume that the second mouth shape is given at the time with a proportion of 80.95% of the total time of the syllable. Taking “ $\text{ㄝ$ ” as an example, we have the result shown in Fig. 5.6.



Fig. 5.6 An illustration of time intervals of two basic mouths.

B. A syllable with three basic mouth shapes.

There are 30 Mandarin finals with 3 basic mouths. By repeatedly doing three times, the experimental results of gathering the first time intervals and the second time intervals of the pronunciations of the basic mouth shapes is illustrated in Fig. 5.7.

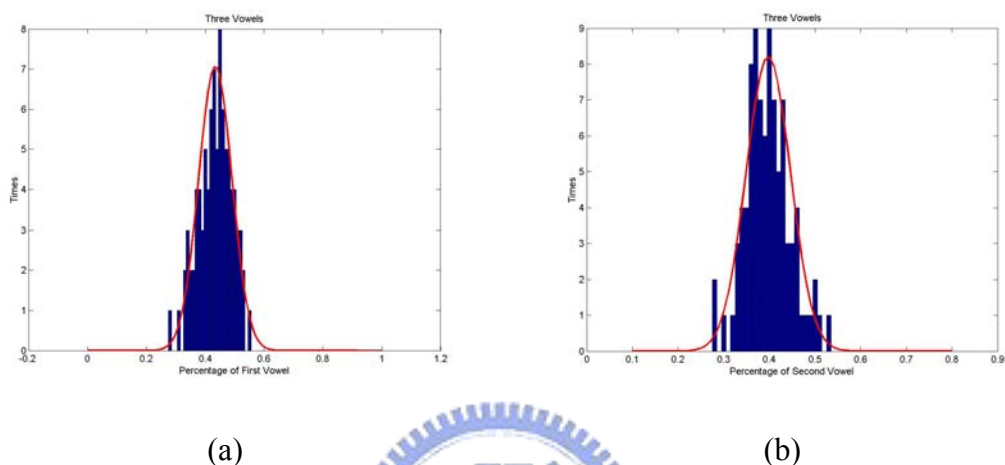


Fig. 5.7 A statistic result of the syllables with 3 basic mouth shapes. (a) The start time of the second basic mouth shape. (b) The time interval between the second and the third basic mouth shape.

From the experimental results of the first time interval between the first and second basic mouth shapes, a normal distribution with $mean = 0.4337$ and $variance = 0.0564$ normalized by the total audio time was obtained. Besides, with the experimental results of the time interval between the second and the third basic mouth shapes, a normal distribution with $mean = 0.3975$ and $variance = 0.0486$ normalized by the remaining audio time is obtained. Therefore, we assume that the second mouth shape is given at the time with a proportion 43.37%, and the third mouth shape is given at the time with a proportion 39.75% of the total syllable time. Taking “一 幺” as an example, we have the result shown in Fig. 5.8.

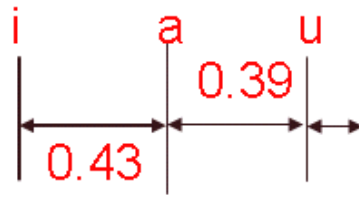


Fig. 5.8 An illustration of time intervals of three basic mouths.

C. A syllable with four basic mouth shapes.

10 syllables with 4 basic mouths were taken for experiments. By repeatedly doing three times, the experimental results of gathering the first time intervals and the second time intervals of the pronunciations of the basic mouth shapes are illustrated in Fig. 5.9.

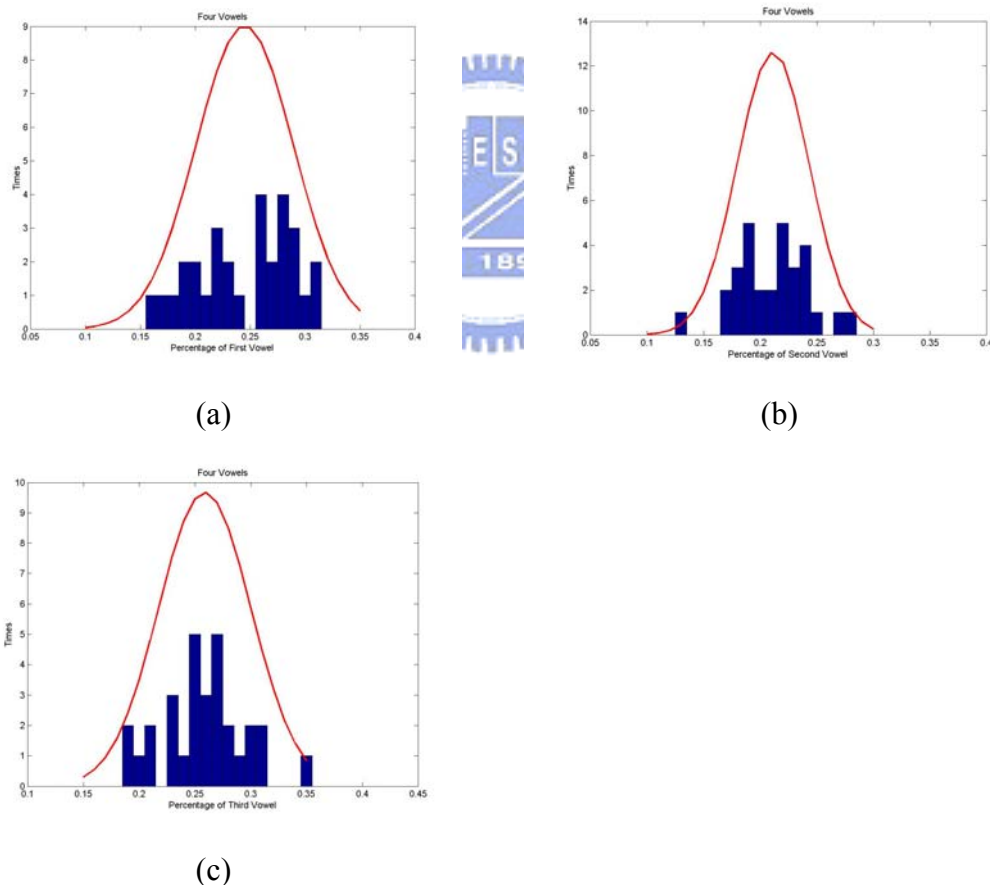


Fig. 5.9 A statistic result of the syllables with 4 basic mouth shapes. (a) The start time of the second basic mouth shape. (b) The time interval between the second and the third basic mouth shape. (c) The time interval between the third and the fourth basic mouth shape.

With the experimental results of the first time intervals between the first and second basic mouth shapes, a normal distribution with $mean = 0.245$ and $variance = 0.0443$ normalized by the total audio time was obtained. With the experimental results of the time intervals between the second and the third basic mouth shapes, a normal distribution with $mean = 0.2115$ and $variance = 0.0316$ normalized by the remaining audio time is obtained. Finally, the experimental results of the time intervals between the third and the fourth basic mouth shapes, a normal distribution with $mean = 0.2588$ and $variance = 0.0412$ normalized by the remaining audio time is obtained.

Therefore, we assume that the second mouth shape is given at the time with a proportion 24.5%, that the third mouth shape is given at the time with a proportion of 21.15%, and that the fourth mouth shape is given at the time with a proportion of 25.88% of the total syllable time. Taking “ m-i-a-u ” as an example, we have the result shown in Fig. 5.10.

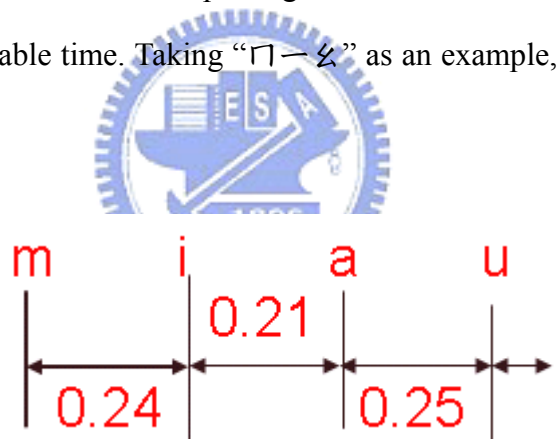


Fig. 5.10 An illustration of time intervals of four basic mouths.

5.4 Talking Cartoon Faces Generation by Synthesizing Moving Lips

An overall illustration of talking cartoon face generation is described in Fig. 5.11.

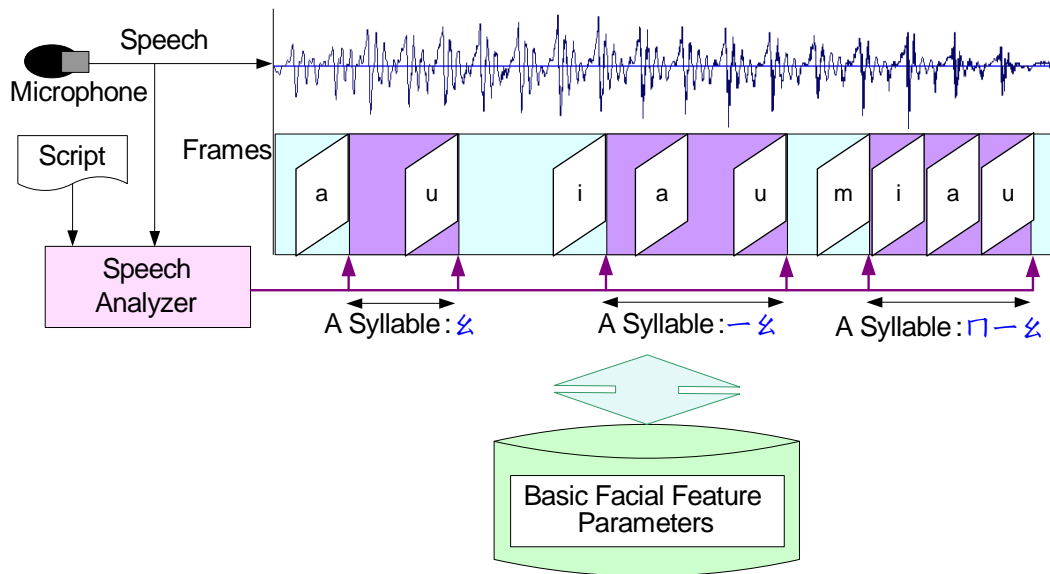


Fig. 5.11 An overall illustration of talking cartoon face generation.

There are two main steps to create a talking cartoon face by synthesizing moving lips after extracting the information of a speech file and a script file by a speech analyzer.

- 1. Obtaining time information for syllables.**

Based on this step, we know when and what to speak. The information of the time of each syllable is obtained for the subsequent use.

- 2. Extracting basic mouth shapes from each syllable and assigning the related parameters to proper key frames.**

Based on this step, we know how to speak. For each syllable, by the previously mentioned method, a combination of basic mouth shapes is obtained. Additionally, by referencing the results of analysis of the time intervals of mouth shapes given in Section 5.3, we know how to assign the basic mouth shapes to proper key-frames.

Finally, by applying an interpolation technique between frames and then synchronizing with the speech file, a talking cartoon face by synthesizing moving lips is created.

5.5 Talking Cartoon Faces Generation by Synthesizing Emotions

Additionally, we can assign the talking cartoon face a freely specified emotion. By reassigning locations to the feature points of eyebrows and eyes, the cartoon faces may appear to be more different.

Some experimental results are given in Fig. 5.12.

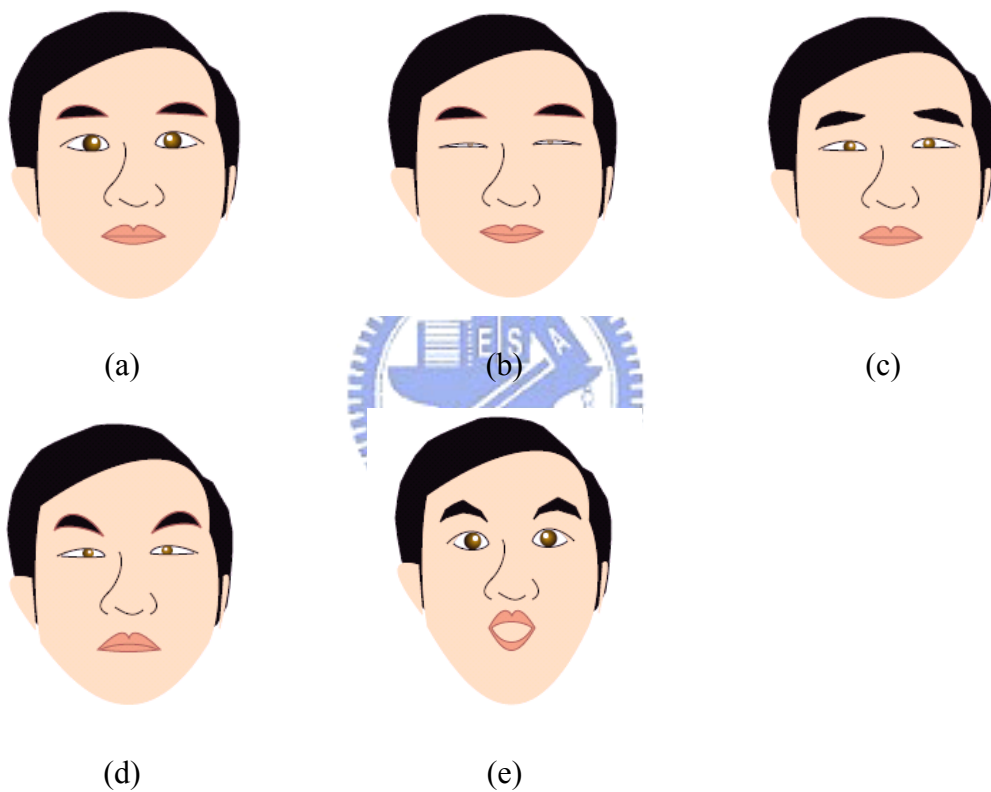


Fig. 5.12 Some experimental results of emotions rendered by SVG. (a) Neutral cartoon face. (b) Smile. (c) Sad. (d) Angry. (e) Surprise.

5.6 Experimental Results

An experimental result of talking cartoon faces is shown here. By applying the

moving lip synthesis technique mentioned previously, a speech file of saying two Mandarin words “您好” is used to create a talking cartoon face. Next, by assigning a “surprise” emotion, a talking cartoon face is obtained as shown in Fig. 5.13.

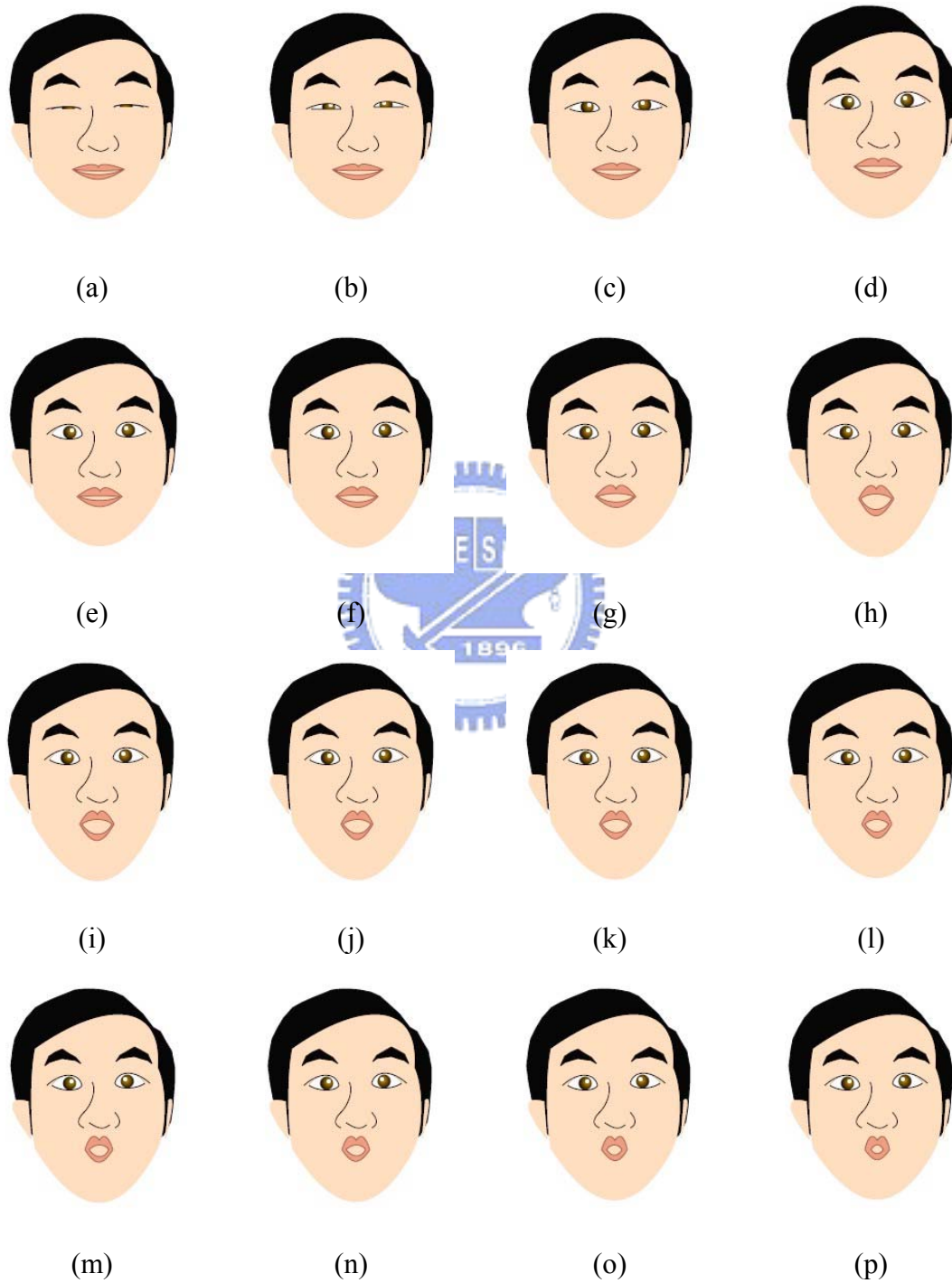


Fig. 5.13 A resulting sequence of the talking cartoon face for speaking “您好” with a surprise emotion and a randomly generated eye-blinking.

Chapter 6

Talking Cartoon Face Generation from Image Sequences

6.1 Introduction

Another way to generate a talking cartoon face is to take a video clip of a user's speaking face as input, and then transform it into a talking cartoon face. A flowchart of talking cartoon face generation from image sequences is shown in Fig. 6.1.

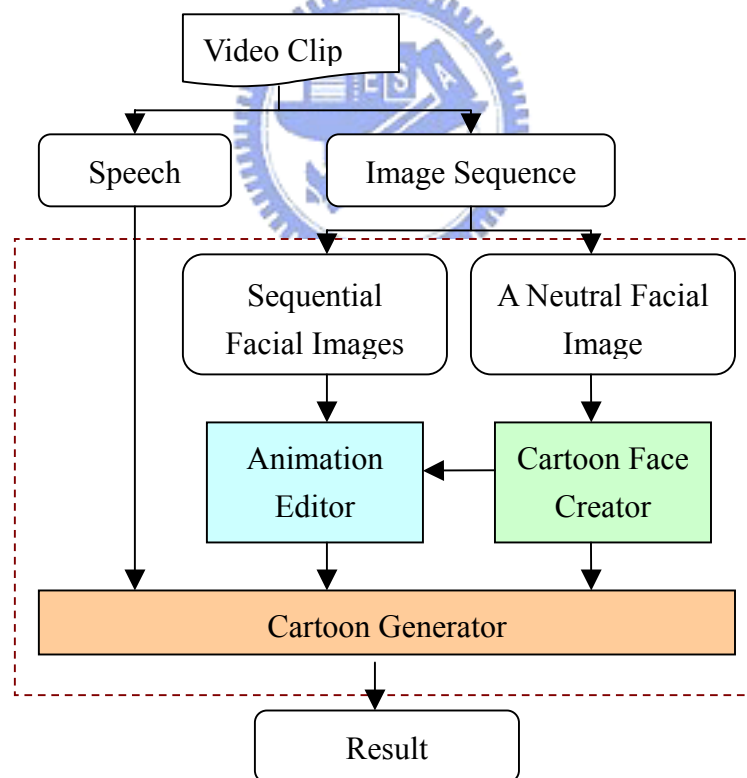


Fig. 6.1 A flowchart of talking cartoon face generation from image sequence.

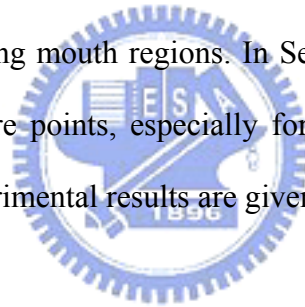
After the proposed system gets an image sequence and a speech file, the cartoon

face generator will automatically choose the first frame as a neutral cartoon face of the user. Some threshold values and parameters will be reserved for subsequent uses.

The animation editor will use the reserved values mentioned previously to process each of the sequential images, and then apply the tracking procedure to effectively detect the desired facial feature regions.

After extracting the desired facial feature region, the method for extracting facial feature points described previously is applied. By using these facial feature points as animation information, the neutral cartoon face can be animated. Since the process of the neutral facial image analysis has been mentioned previously, here we only focus on how to track the demanded facial feature points.

In Section 6.2, a method for tracking eyeball regions is described. Section 6.3 describes a method for tracking mouth regions. In Section 6.4, an extraction method for extraction of facial feature points, especially for extraction of control points is described. Finally, some experimental results are given in Section 6.5.



6.2 Tracking for Eyeball Regions

In the proposed system, a method for tracking eye regions is proposed. The main idea of the method is described as follows:

1. Reserve a threshold value t_1 and a pair of eyeball regions $Eyeball_{Left}$ and $Eyeball_{Right}$ during the creation of the neutral cartoon face for further uses.
2. Use t_1 for the 1st level hierarchical bi-level thresholding in the intensity channel to get a binary image B_1' .
3. Use the pixels of $Eyeball_{Left}$ and $Eyeball_{Right}$ as seeds and applying region growing on B_1' to extract some possible eye regions.

4. Use the eye-pair detection method mentioned in Chapter 3 to get a new pair of eye regions. If no eye-pair is detected, use the old pair of eye regions to substitute new ones.
5. Use the eyeball detection method to get new eyeballs $Eyeball_{Left}$ and $Eyeball_{Right}$.
6. Reserve the new eye-pair regions for eye-pair tracking in the next facial image and go to Step 2 until the last image of the video sequence has been dealt with.

A detailed illustration of the above procedure is shown in Fig. 6.2

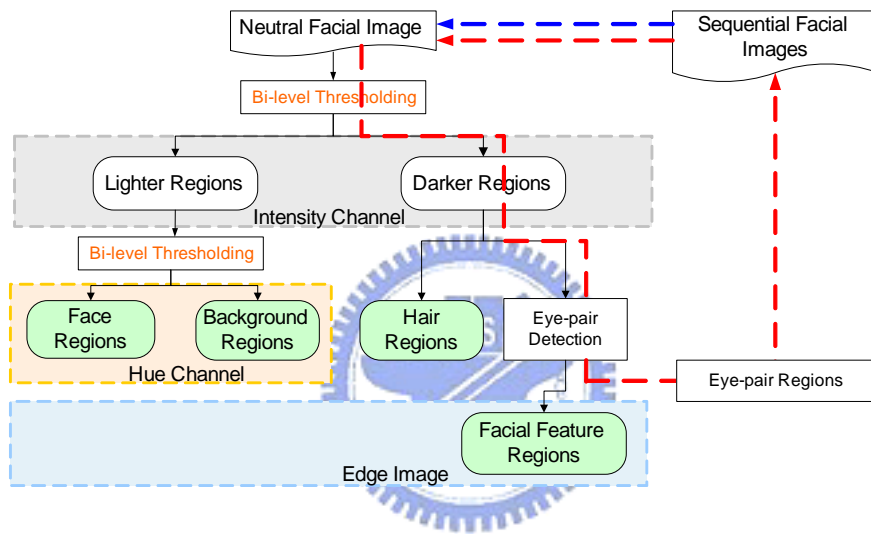


Fig. 6.2 A flowchart of tracking for eyeball regions in image sequences.

6.3 Tracking Mouth Regions

In the proposed system, a method for tracking mouth regions is proposed. The main idea of the method is described as follows:

1. Reserve the three horizontal division lines, Div_{EE} , Div_{EN} , Div_{NM} , the four threshold values of four parts of the edge image, and a mouth region R_{Mouth} computed during the creation of the neutral cartoon face for further uses.

2. Use the parameters mentioned in Step1 to threshold the *Sobel* edge image of each subsequent facial image.
3. Use the pixels of R_{Mouth} as seeds, and apply region growing to extract a new mouth region.
4. Go to Step 2 until the last frame of the sequential image has been processed.

A detailed illustration is shown in Fig. 6.3.

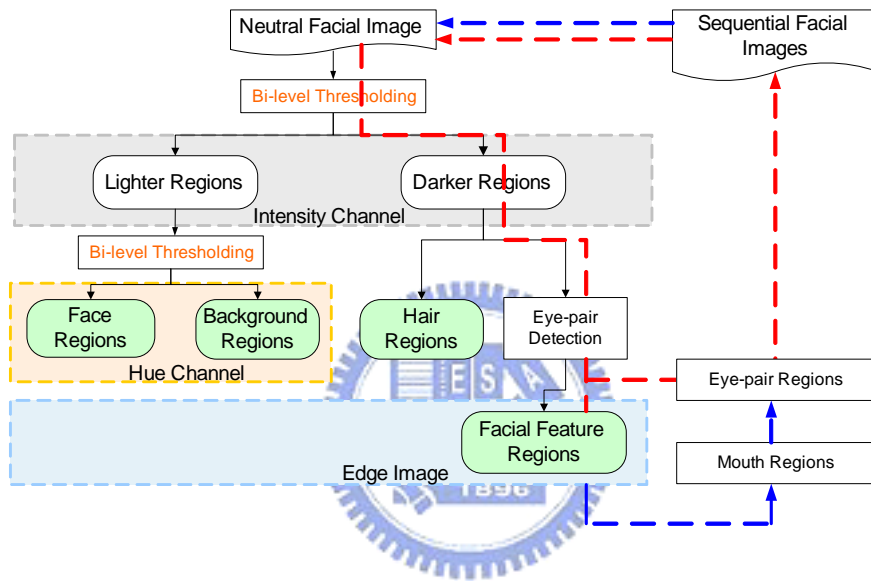


Fig. 6.3 A flowchart of tracking for Mouth regions in image sequences.

6.4 Extraction of Facial Feature Points

After extracting eyeball regions and mouth regions, a method of region refinement mentioned previously is applied. Next, a method for extraction of facial feature points mentioned previously is use to extract eye points and mouth points, especially to detect control points for animation. The tracking results of the sizes of mouth shapes need be adjusted with the nine basic mouth shapes.

6.5 Experimental Results

In this chapter, a method for tracking facial features to accomplish face animation through image sequences is proposed. Some cartoon faces were animated by successful image tracking results. An experimental result of the extracted animation information and the corresponding animated cartoon faces are shown in Fig. 6.4 and Fig. 6.5. Another experimental result is shown in Fig. 6.6.

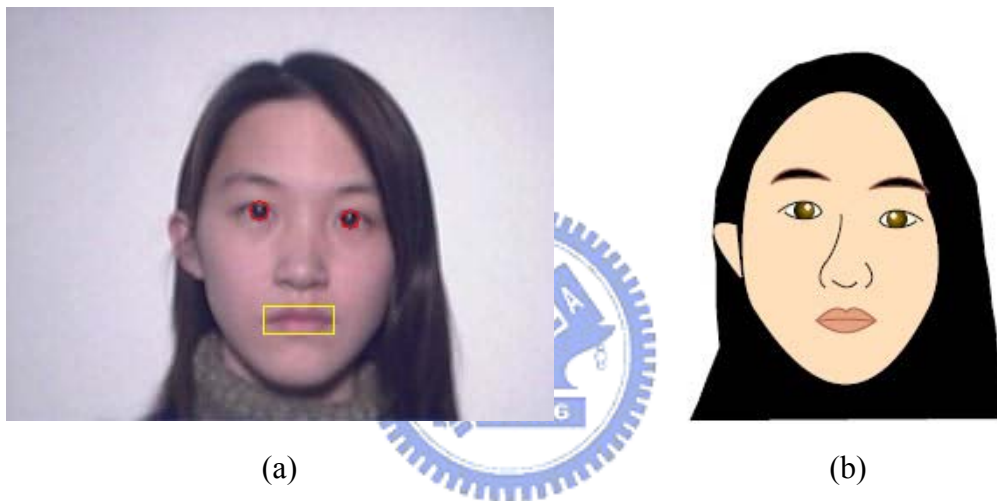
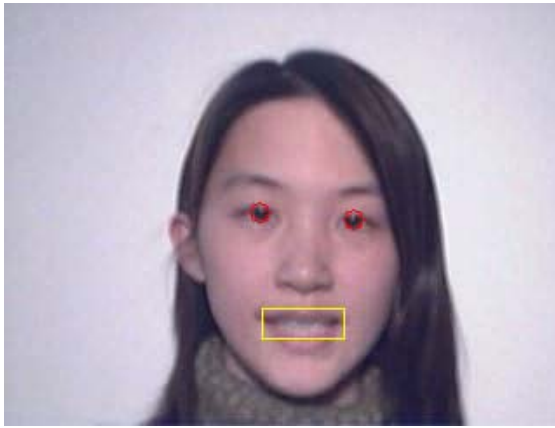
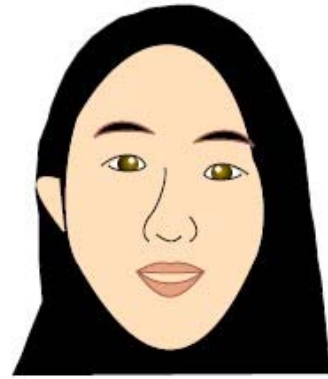


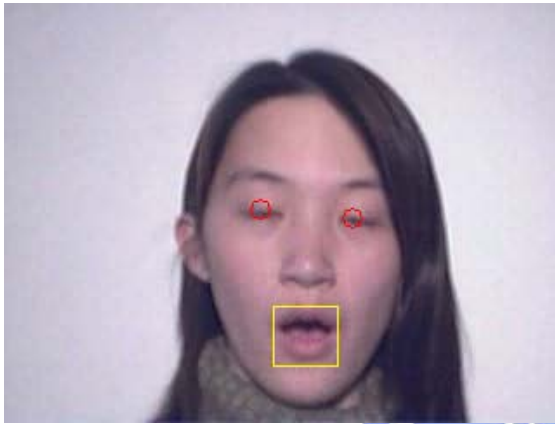
Fig. 6.4 An example of experimental results. (a) A given neutral facial image (the first facial image of the video sequence). (b) A result of automatic neutral cartoon faces generation.



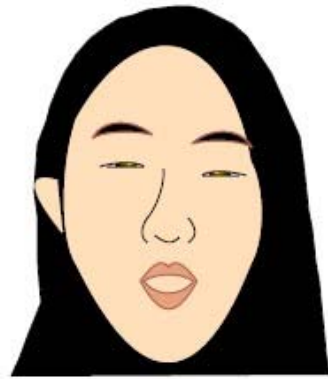
(a)



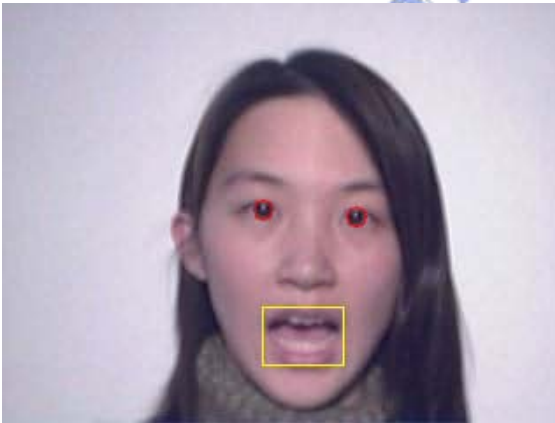
(b)



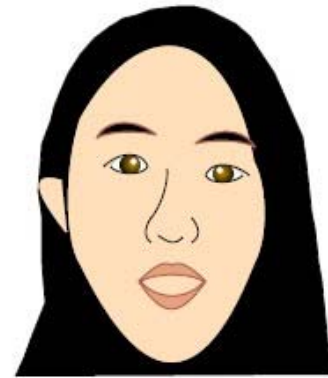
(c)



(e)



(e)



(f)

Fig. 6.5 An example of tracking results. (a) A tracking result of one of the image sequence. (b) Related cartoon face of (a). (c) A tracking result of one of the image sequence. (d) Related cartoon face of (c). (e) A tracking result of one of the image sequence. (f) Related cartoon face of (e).

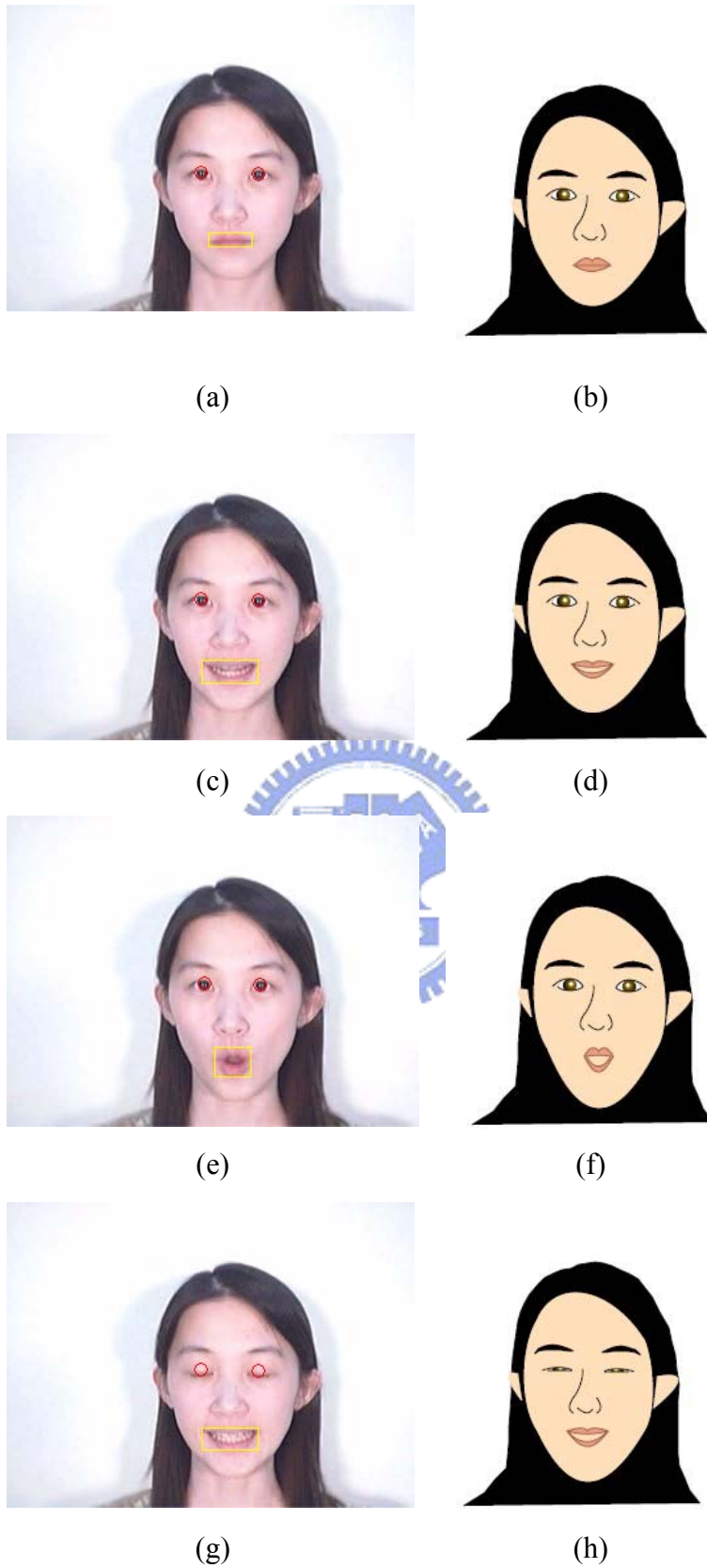


Fig. 6.6 Another experimental result. (a) A given neutral facial image (the first facial image of the video sequence). (b) A result of automatic neutral cartoon faces generation. (c)(e)(g)

Chapter 7

Cartoon Generator and Experimental Results

7.1 Introduction

In the proposed system, the generated talking cartoon faces are rendered by an application language SVG (Scalable Vector Graphics), which is an editable and opened vector-based XML language of W3C (World Wide Web Consortium). The reasons why we use it in this study are described in Section 7.2. In Section 7.3, a more detailed method for using the SVG language to display talking cartoon faces is described. Finally, some applications are described and shown in Section 7.4.

7.2 Overview of SVG

SVG is a language for describing two-dimensional vector and mixed vector/raster graphics in XML. It allows three types of graphic objects: vector graphic shapes, images (including JPEG, PNG, and GIF, and so on), and text. Each of them can be grouped, styled, transformed, and so on. SVG also supports some common effects on objects, including alpha mask, clipping, etc. It also has the capability for interacting with people and for more abundant applications when combined with other

languages, including HTML, JavaScript, SMIL, and so on. Additionally, SVG drawings are vector-based; they can be freely scaled without destroying their quality. It is very suitable for applications involving virtual cartoon faces.

There are still some other main advantages of using SVG in the proposed system, as listed in the following:

1. It supports the two curve drawing methods used in the proposed system. We can draw a cartoon face simply by applying the related syntax provided by it.
2. The source codes of SVG are readable, editable, and searchable. It can be easily created without using specific software.
3. SVG is an open standard. There are already some tools for creating, playing, and converting SVG to certain other file formats. It is convenient to use it for general applications.
4. Last but not the least, SVG drawings can always be created with less data for easier sharing.



A simple example showing a line of styled text is shown in the following:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN"
"http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg width="400px" height="250px">
  <text x="50px" y="30px" style="stroke:blue; fill:red; font-size:18" >
    Every SVG document has an &lt;svg&gt; element.
  </text>
</svg>
```

The result of the above source code is shown in Fig 7.1.

Every SVG document has an `<svg>` element.

Fig. 7.1 A result of an SVG source code.

Since the source code of SVG is meaningful and editable, we just have to write the desired code into an SVG file. We can directly view the SVG drawing by an IE browser.

7.3 SVG Animation

In order to get a desired talking cartoon face, two kinds of views are applied for rendering. One is the way from the view in the spatial domain and the other is from the view in the temporal domain, both being described in Section 7.3.1 and 7.3.2, respectively.

7.3.1 From the View of Spatial Domain

From the view of the spatial domain, the main goal is to draw a cartoon with the concept of layers using the SVG language. A concept of the spatial domain is shown in Fig. 7.2.

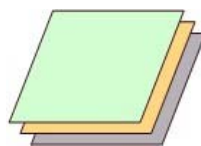


Fig. 7.2 An illustration of spatial domain.

A neutral cartoon face can be drawn by the syntax provided by SVG, including path (including cubic Bezier curve), circle, polyline, etc. In Chapter 4, a face model with facial feature points was mentioned. A cartoon face is obtained by assigning the locations of feature points as parameters to specified syntax.

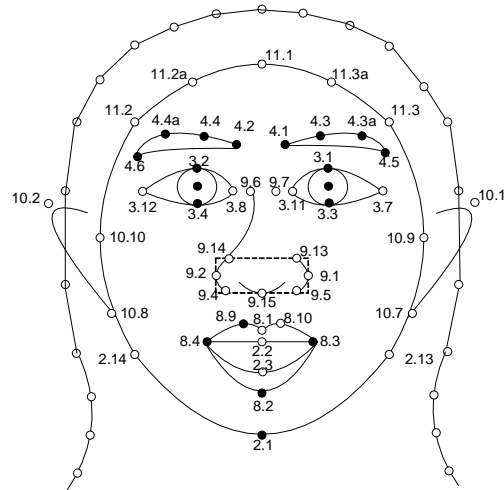


Fig. 7.3 A proposed face model with 84 facial feature points.

In the proposed system, “polyline”, which describes a shape created by a number of straight lines, is used to draw the hair with the hair points as parameters to it. It is also used for eyebrow drawings.

```
<polyline points="30,200 30,50 130 50 30,200"
style="stroke:#000000; fill:#FF0000; stroke-width:2" />
```

As seen above, we can specify the shape of the polyline by assigning demanded points. The style of the polyline can even more be modified by specifying the parameters of the style.

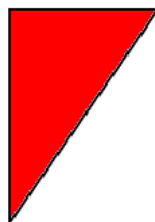


Fig. 7.4 An illustration of the polyline shape.

Next, “circle” is used for eyeball drawings. Since the eyeball will be concealed during eye-blinking, a special effect of clipping is applied here to solve the problem. The basic syntax of “circle” is described as follows:

```
<circle cx="100" cy="100" r="50" style="stroke:#FF0000; fill:#FF0000; stroke-width:2 " />
```

As seen above, we can specify the shape of the circle by assigning the position of the center of a circled and the radius. The style of the circle can even more be modified by specifying the parameters of the style.

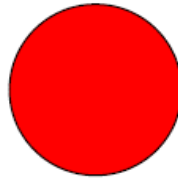


Fig. 7.5 An illustration of the circle shape.

A rendering result of eyes is shown in Fig. 7.66. The remaining facial features are drawn by applying the syntax of the cubic Bezier curve, as shown in Fig. 7.7.

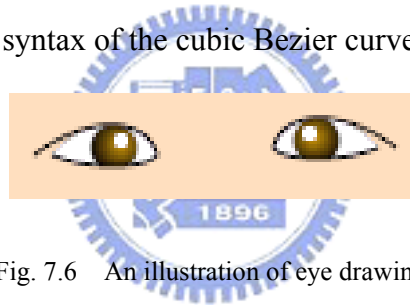


Fig. 7.6 An illustration of eye drawing.

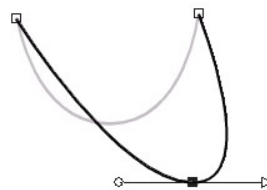


Fig. 7.7 An illustration of the cubic Bezier curve.

We can also specify the colors of each component. Thus, we can modify the cartoon faces by reassigning the shapes and styles of the svg drawings. Fig 7.8(a) shows a default rendering style of a cartoon face. Fig 7.8(b) uses cubic curve to draw the eyebrows. A cute type cartoon face is created by modify a jaw shape, two eyeball shapes, and a nose shape by a predefined rule.

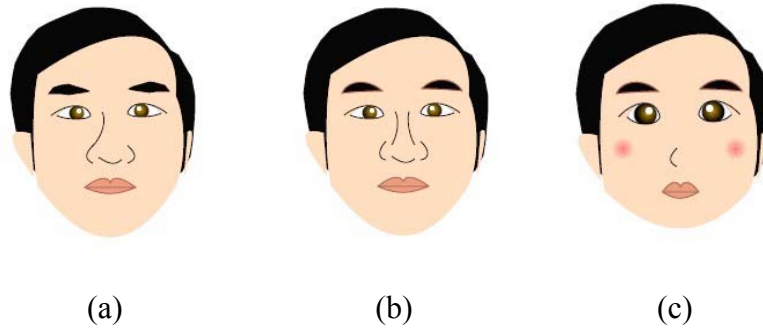


Fig. 7.8 A result of different types of cartoon face. (a) Normal type. (b) Different types of eyebrows and nose. (c) Cute type.

There are two main abstract layers in the proposed system: the static layer and the dynamic layer. The static layer at the back is used to draw the components which are existing all the time. On the contrary, the dynamic layer is used to draw animation components.

Besides, we can add one or more layers to draw something else.



Fig. 7.9 A result of adding a layer for clothing with cute-type faces.

7.3.2 From the View of Temporal Domain

From the view of temporal domain, the main goal is to synchronize the speech file with the animations using the SVG language.

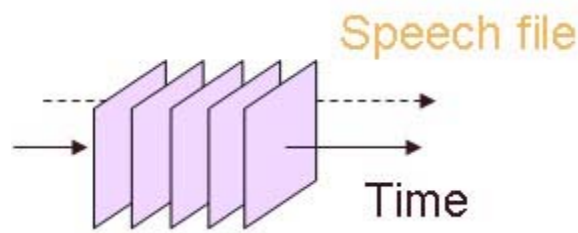


Fig. 7.10 An illustration of temporal domain.

A. Speech file playing

The syntax for audio playing provided by the application language SVG is shown in the following.

```
<a:audio xlink:href="young.wav" begin="0s"/>
```

As seen above, we can specify when the audio file starts to play.

B. Animation simulation

In the proposed system, a kind of animation is to simulate a frame sequence by the following way:

```
<g visibility="hidden">
  <set attributeName="visibility" from="hidden" to="visible"
    begin="0.000000s" dur="0.040000s"/>
    ..Components of a frame..
</g>
```

As seen above, we can use “g” to group a set of facial components which belong to a frame, and then specify the properties of time and visibility to simulate animation by the concept of frame rate.

Since each frame can be animated, we can use this property to shake the head, as shown in Fig. 7.11.



Fig. 7.11 Cute type cartoon face with head shaking animation.

7.4 An Application to Electronic Books

The generated talking cartoon faces can be viewed through the IE or Netscape browser by installing a plug-in. An application is shown in Fig. 7.12.

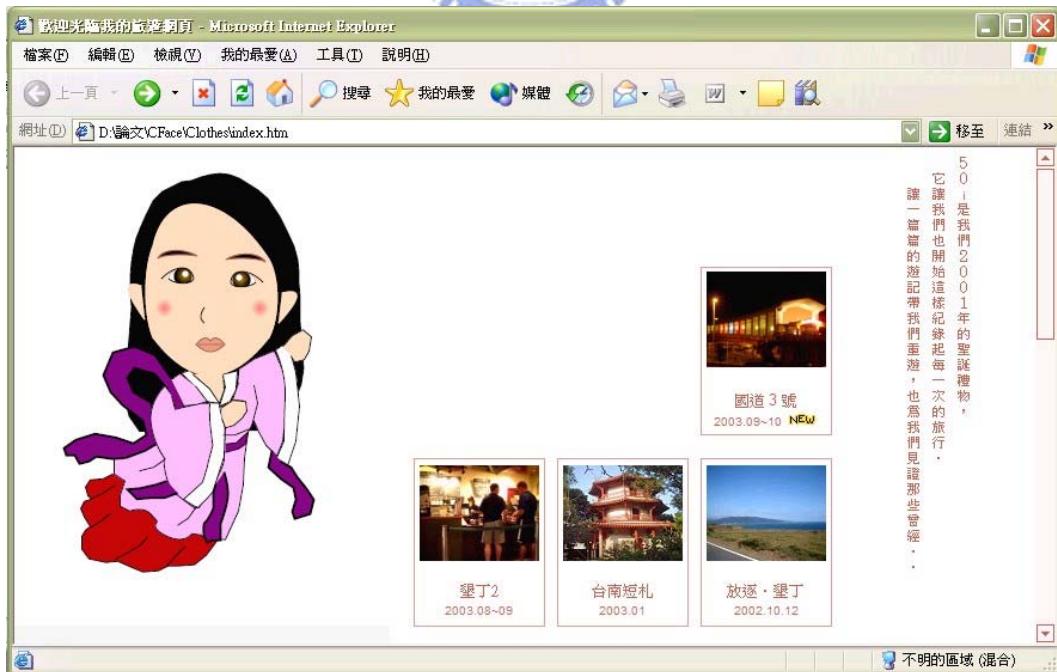


Fig. 7.12 An application of proposed methods to electronic books.

7.5 Experimental Results

In this study, talking cartoon faces can be generated from single images or sequential images. First, we use a single facial image as input to generate a talking cartoon face. The detailed method is described in Chapter 5. The input neutral facial image used here is shown in Fig. 7.13. The way to animate the cartoon face is by synthesizing moving lips from a given speech of two Mandarin words “您好”. An experimental result is shown in Fig. 7.15.



Fig. 7.13 The neutral facial image used for the experiment.

Another experiment is using a video clip with a user’s speaking face as input to generate an animation of a talking cartoon face. The detailed method for use in this experiment is described in Chapter 6. The proposed system takes the first frame of the video sequence to generate a cartoon face, and then animates the face by facial features tracked in the image sequence. The video sequence is shown in Fig. 7.14. The user was saying two Mandarin words “夕陽”. An experimental result is shown in Fig. 7.16.

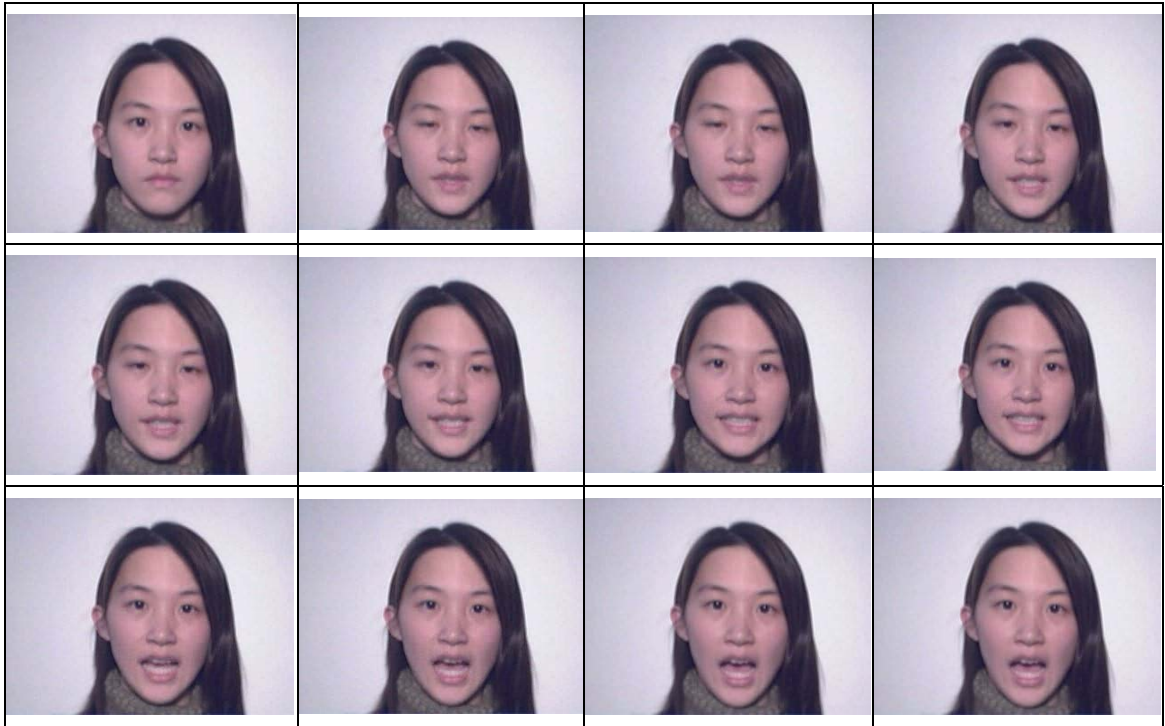


Fig. 7.14 A video sequence used for the experiment.



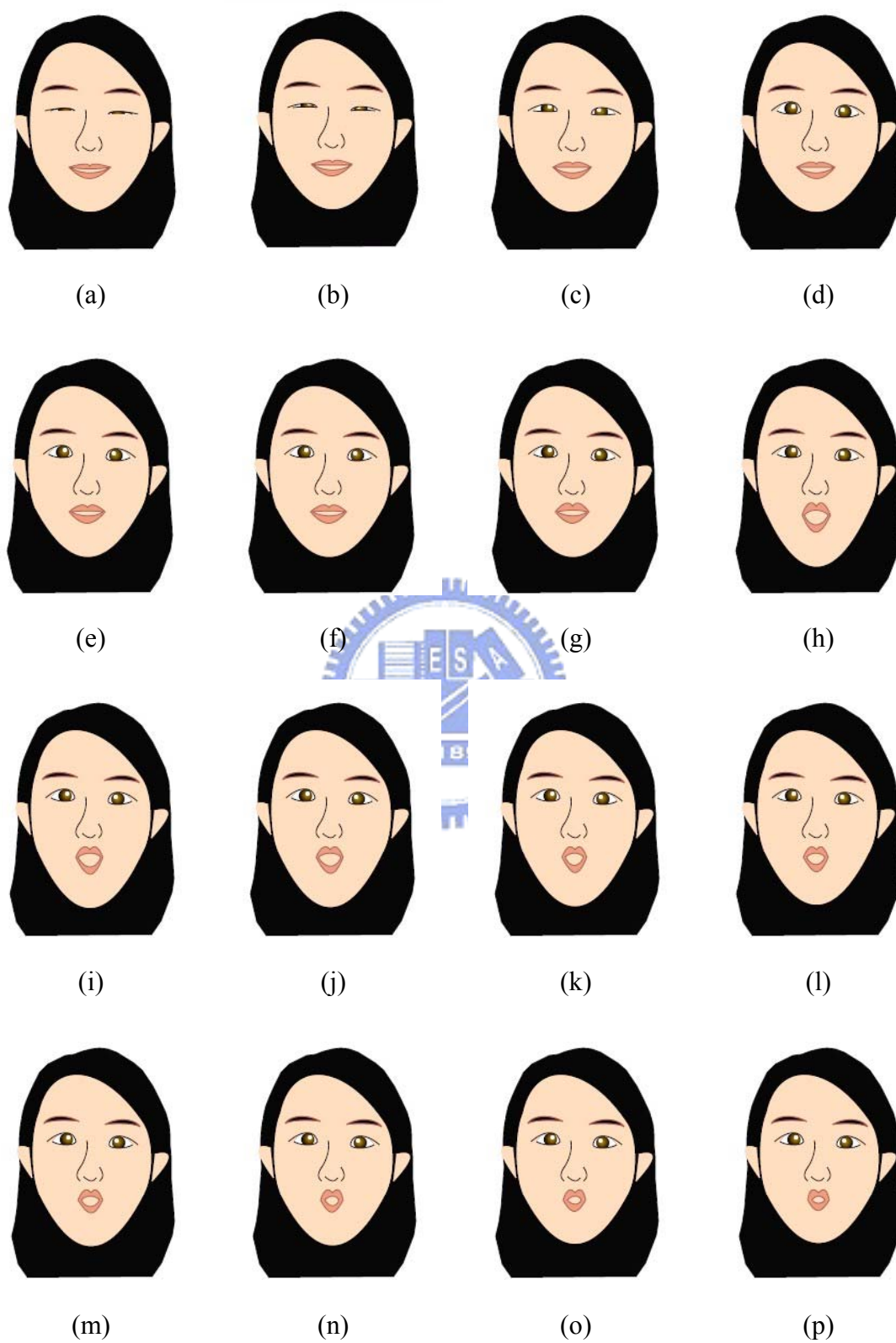


Fig. 7.15 A resulting sequence of the talking cartoon face for “您好” with a random generated eye-blinking.

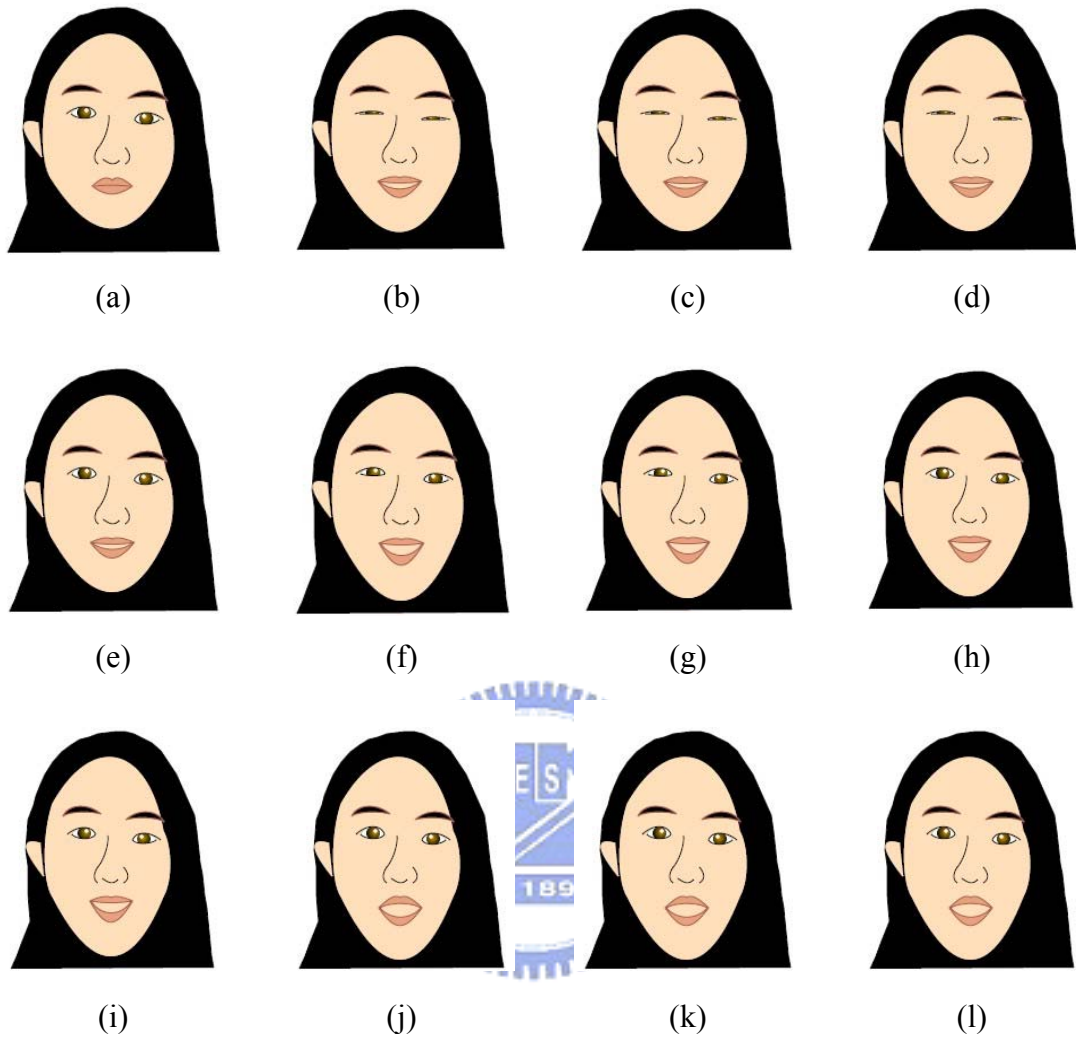


Fig. 7.16 Another experimental results of talking cartoon face generation from image sequences, where the speaker in the video clip is speaking “夕陽”.

Chapter 8

Conclusions and Suggestions for Future Works

8.1 Conclusions

In this study, a system for automatic generation of talking cartoon faces has been implemented. We have presented a way to automatically create a personal cartoon face from a given neutral facial image, and then animate it by moving-lip synthesis and facial feature tracking. The system consists of three components, including a cartoon face creator, an animation editor, and a cartoon generator.

By the cartoon face creator, facial feature regions and facial feature points can be extracted automatically from a neutral facial image to create a personal cartoon face. This is achieved by the use of a hierarchical bi-level thresholding method and a knowledge-based image analysis technique proposed in this study. A face model of 72 facial feature points is proposed for cartoon face drawing. Dragging these feature points can directly deform the cartoon face. Several of these facial feature points are assigned to be control points for easier animation of cartoon faces.

Next, using the animation editor, animation information can be extracted from an image sequence to make the cartoon face look like the person in the image sequence. Cartoon faces can also be animated by the use of sequential facial images through automatic tracking of the facial features. Animation information can also be extracted from a speech and a script. A method for translating each syllabus into a combination

of 9 basic mouth shapes is proposed. A syllabus may be composed of one to four basic mouth shapes. Furthermore, relations between mouth shapes are also analyzed in this study. A moving-lip synthesis technique for cartoon face animation is thus proposed.

Finally, in the component of the cartoon generator, an editable and opened vector-based XML language of W3C (World Wide Web consortium) standard, namely, the SVG (Scalable Vector Graphics), is used for rendering and synchronizing the cartoon face with speech. The outlook of the cartoon face can be specified in this component.

The experimental results shown in the previous chapters have revealed the feasibility of the proposed system.

8.2 Suggestions for Future Works



Several suggestions to improve the proposed system are listed as follows.

- (1) Improvement on facial feature detection--- In order to demonstrate low-cost processing, the facial feature detection method need be improved to fit more application environments.
- (2) Improvement on similarity measurement between users and generated cartoon faces --- Detections of glasses, wrinkles, and hair styles of given facial images can be improved. More personal mouth shapes can be used for synthesizing moving lips.
- (3) Improvement on animations of human facial expressions --- If a user chooses a video clip as input, he may want the generated cartoon faces act more like him/her. A more delicate tracking of facial features can improve the satisfaction of more people. Even more, the speech file can be analyzed by the given

technique proposed in this study to improve the result of talking cartoon faces.

- (4) Real-time animation generation --- By using the existing techniques for facial feature tracking of sequential images, there exists a potential for real-time animation generation.
- (5) Rendering cartoon faces with more types --- In the component of the cartoon generator, more cartoon types can be supported for face rendering to raise the quality of the resulting cartoon faces. Moreover, a way to learn a cartoonist's cartoon style would make the cartoon face more attractive for more people.



References

- [1] Y. C. Lin and W. H. Tsai, "A Study on Virtual Talking Head Animation by 2D Image Analysis and Voice Synchronization Techniques," *M. S. Thesis*, Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan, Republic of China, June 2002.
- [2] W. H. Tsai, "Moment-preserving thresholding: a new approach," *Computer Vision, Graphics, and Image Processing*, 1985, vol. 29, pp. 377-393.
- [3] Y. S. Chen, C. H. Su, J. H. Chen, C. S. Chen, Y. P. Hung, C. S. Fuh, "Video-Based Eye Tracking for Autostereoscopic Displays," *Optical Engineering*, Dec. 2001, vol. 40, no. 12, pp. 2726-2734.
- [4] Z. Ruttkay, H. Noot, "Animated CharToon faces," *Proceedings of the 1st international symposium on Non-photorealistic animation and rendering*, Annecy, France, June 05-07, 2000 pp.91-100.
- [5] G. J. Edwards, A. Lanitis, C. J. Taylor, T. F. Cootes, "Face recognition using statistical models," *Image Processing for Security Applications, IEE Colloquium on Image Processing for Security Applications*, UK, Mar. 10, 1997, pp. 2/1-2/6.
- [6] Rein-Lien Hsu, A. K. Jain, "Generating discriminating cartoon faces using interacting snakes," *IEEE Transactions on Pattern Analysis and Machine Intelligent*, 2003, vol. 25, pp. 1388 -1398.
- [7] P. Litwinowicz and L. Williams. "Animating images with drawings," *SIGGRAPH94 Conference Proceeding*, Addison Wesley, August 1996, pp. 225-236.
- [8] H. Chen, N. N. Zheng, L. Liang, Y. Li, Y. Q. Xu, H. Y. Shum, "PicToon: a personalized image-based cartoon system," *Proceedings of the tenth ACM*

- international conference on Multimedia*, France, 2002, pp.171-178.
- [9] Y. Li, F. Yu, Y. Q. Xu, E. Chang, H. Y. Shum, “Speech Driven Cartoon Animation with Emotions,” *Proceedings of the ninth ACM international conference on Multimedia*, 2001, pp. 365 – 371.
- [10] S. C. Y. Chan , P. H. Lewis, “A Pre-filter Enabling Fast Frontal Face Detection,” *Proceedings of the Third International Conference on Visual Information and Information Systems*, June 02-04, 1999, pp.777-784.
- [11] I. Grinias, Y. Mavrikakis, G. Tziritas, “Region Growing Colour Image Segmentation Applied to Face Detection,” *Intern. Workshop on Very Low Bitrate Video Coding*, 2001.
- [12] J. Ostermann, “Animation of Synthetic Faces in MPEG-4,” *Proceedings of the Computer Animation*, June 08-10, 1998, p.49.
- [13] T. Goto, M. Escher, C. Zanardi, N. Magnenat-Thalmann, “MPEG-4 based animation with face feature tracking,” *Proc. Erographics Workshop on Computer Animation and Simulation'99*, Milano, Italy, September. 7-8 1999, Springer, Wien New York, pp.89-98.
- [14] D. Burford, E. Blake, “Real-time facial animation for avatars in collaborative virtual environments,” *Proceedings of South African Telecommunications Networks and Applications Conference '99*, 1999, pp. 178-183.
- [15] Guenter, C. Grimm, D. Wolf, H. Malvar, F. Pighin, “Making faces,” *Computer Graphics Proceedings SIGGRAPH'98*, 1998, pp. 55-66.
- [16] T. M. Yeh, *Drills and Exercises in Mandarin Pronunciation*, National Taiwan Normal University, ROC, May 1982.